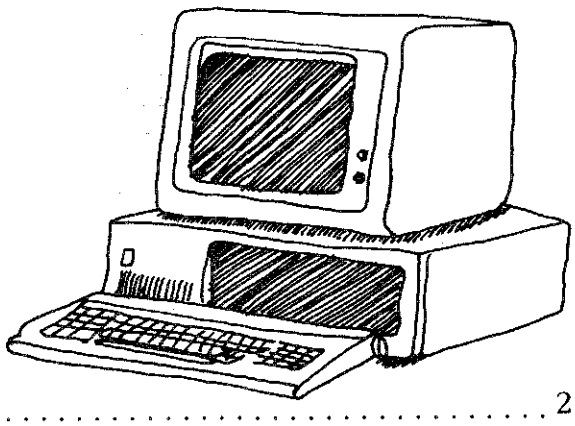
programmer's journal

The Resource Journal for IBM PC Programmers

ASM, C, BASIC, PASCAL

Volume 1 Number 5

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



CONTENTS

HOUSEKEEPING 2

LETTERS 3

PJ PROFILE: Interview with Brett Salter 5
Marketing your own software

LEGAL: AN AGREEMENT OUTLINE / Robert Dunaway 9
Be sure you cover the whole picture

PJ FORUM: THE COPY PROTECTION WAR / Robert T. McQuaid 11
The history and future of copy protection

MULTI-TASKING / Frankie Detesto 13
A look at the Multi products

PASCAL: A CHECK BALANCING PROGRAM / Corey Morrell 15
Structured Pascal programming

C: TRANSPORTABILITY PROBLEMS / Ken Smolek 18
The difficulties of various compilers

ASM: 5 WAYS TO COMBINE MODULES / Art Zachai 20
Connecting segments and modules

BASIC: STRUCTURED PROGRAMMING (Part 2) / Joe Rayhawk 23
Logical approaches to BASIC coding

BIOS EQUATES / from various Bulletin Boards 26
Setting up what you want to find

TOOLS: THE PROGRAMMER'S CALCULATOR / Ron Moak 29
The ultimate online calculator

dBASE II: BARGRAPHS / Dan Ridder 30
Using dBASE II to graph your data

UTILITIES: PROKEY / Bill Hailey 34

SOFTWARE REVIEWS
EDIX / Morton F. Kaplon 35
BASIC DEVELOPMENT SYSTEM / Ron Moak 37

PJ LIBRARY 38

MISC. 40

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOUSEKEEPING

HOUSEKEEPING

There are several new products for the IBM PC that are of particular interest. TRACE86 from Morgan computing is an ASM development and debug tool that does for assembly programming what the Norton Utilities did for understanding diskettes. Princeton Graphic Systems, besides their very popular color monitor, have introduced an AMBER monitor that replaces the standard IBM monochrome. The same resolution exactly. So if your eyes are going on the blink and you wish to try an amber screen without giving up the clarity, you are finally in luck. Another offering from Morgan Computing is a true 16 bit BASIC that will use all of your available memory, the 8087 chip if you wish, extensive execution tracing capabilities, very large array sizes, etc. It is called PROFESSIONAL BASIC, and was written by Neil Bennett.

The new multi-tasking programs are beginning to show up. Their endless possibilities are fettered only by the sheer complexity of providing a coherent bug-free product. I am not sure whether writing a full blown word processor or creating a multi-tasking package running under DOS would be the greater challenge. No doubt it would depend on what features you provided.

The mouse technology is expanding enormously.

I would like to tell you about a humorous incident that occurred at the SF Faire this August. One that we are all familiar with in other contexts. I stopped by a booth which was demonstrating their new MOUSE operating system/wordprocessor. As I was talking to the programmer who wrote part of it (a very sharp fellow), two other 'customers'' showed up to look at the system. The new arrivals did not have on their badges and proceeded to take the Mouse in hand and run it through its paces. It took my programming friend and I about 30 seconds to realize that these two were also software developers because only another hacker knows how to IMMEDIATELY go to the boundary conditions of a program and see how well they have been handled. Like run the mouse off of the screen at all four corners, spin the ball and check both the mechanical reaction to this as well as the programs response, and so on. My friend asked them ''have you used a mouse before?'' and they replied, ''Maybe'' . Then he asked (we were all smiling at this point) ''Have you done any programming for a Mouse?'' , and they replied ''Quite possibly'' . The badgeless duo continued testing the system for another 10 minutes and complimented my friend on his Mouse system and the accurate response to its manipulation.

We all check out the competition. Badgeless or otherwise. But you know the best part of the story? Those two visitors were one of the few conventioners that could really appreciate the programmers work. And even though they may have been from the competition, there was a mutual respect and enjoyment of each others work.

Such is the fraternal world of software developers.

— Greg

Editor/Publisher: Greg Estes
Associate Editor: Frankie Detesto
Contributing Editor: Greg Roberts
Art Director: Julia King
Production: Cassandra Estes
Copy Editing: Renee Anderson

Contributing Writers:
Joe Rayhawk
Arthur Zachai
Ron Moak
Morton F. Kaplon
Dan Ridder
Bill Hailey
Robert T. McQuaid
Robert Dunaway
Corey Morrell
Marilyn & John Walker
Ken Smolek

Editorial and Business Office:
2765 Potter St.
Eugene, OR 97405

Mailing Address:
P.O. Box 5070, Eugene, OR 97405

Programmer's Journal. Volume 1, Number 5.
Copyright (C) 1983 by Programmer's Journal
Published bi-monthly.

Postmaster send address changes to:
Programmer's Journal
2765 Potter St.
Eugene, OR 97405
ph. (503) 687-0398

Subscriptions: Bi-monthly rates are $24 U.S.
($30 in Canada, and $40 elsewhere.)

IBM and Personal Computer are trademarks of International Business Machines, Armonk, New York
dBASE II is a trademark of Ashton-Tate.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

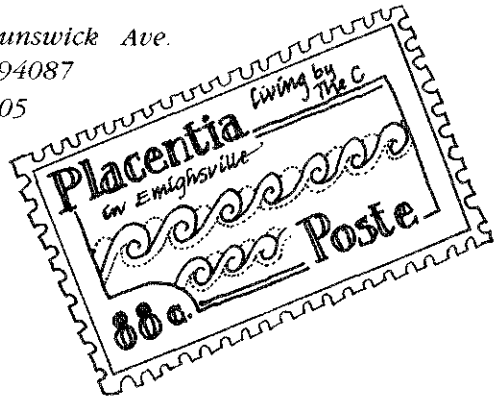
LETTERS

DeSMET 88

By the way, I am using the DeSmet C88 Compiler which is distributed by C-Ware in Sunnyvale. I have been singing its praises in my IBM club newsletter, and to anyone else who will listen since I first ordered it. It is an outstanding development package, but truly amazing for the price of $100! That full price includes not only the compiler, but a full screen editor, an assembler, a linker, and an extensive set of standard and non-standard library functions. You will notice that the new August issue of BYTE is devoted almost entirely to the C language. DeSmet C is compared against 5 others for CP/M and against 8 others in the PC-DOS versions. In each case it gets very high marks, holding its own against the best $500 glossy-ad hot shots. The word should be spread.

Glenn Emigh
Placentia, CA

DeSmet C
C Ware
1607 New Brunswick Ave.
Sunnyvale, CA 94087
ph. 408-736-6905



HARD DISKS

I have heard rumors that Radio Shack is preparing an IBM clone ($1500), as well as Hewlett-Packard. I also considered getting the Great Lakes 10 meg hard disk for $995. Have since heard they are having problems. 1) Doesn't work with old PC's. 2) Supplier changed controller board without notifying them, etc. 3) Also heard they were considering getting out of mail order and were raising their prices. Interesting world! Also not all ramdisks work with all hard disks.

Also the order in which PROKEY is installed seems to be important if you are using other drivers.

Sincerely yours,

Bill Hailey
Houston, TX

BASIC 2.0 PRTSC

I have discovered an interesting tip that you may want to pass along to your readers. Using IBM PC BASIC 2.00, it is possible to duplicate the function of the Ctrl-PrtSc key combination under program control.

To turn ON echo of the screen output to the printer:

```
DEF SEG : POKE &H758,&HFF
```

To turn this printer echo OFF:

```
DEF SEG : POKE &H758,0
```

Your truly,

Ken Snapp
Softool Systems
Denver, CO

IBM BOCA TIPS
EPSON PRINTER TIP

When using the printer to print your Macro Assembler .LST files, put the following line at the beginning of your .ASM files to automatically switch the printer to 132 column mode:

```
PAGE ,132 ;^O
```

That's a Control-O seen by the printer but ignored by the assembler. To automatically revert to 80-column mode at the end of the listing, append a comment with a Control-R to the last line in your listing:

```
.... ;^R
```

USING RADIX 16

If you use .RADIX 16 to avoid typing in H's after all your hex numbers, it won't work for hex 0B or 0D. This is because the assembler will always interpret them as meaning the no. is Binary (B) or Decimal (D) - unless they are followed by an H, i.e. 0BH.

Thanks to Boca Raton for these tips.

JW

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PMATE REVISITED

To characterize PMATE, we find that PMATE is fast; PMATE is powerful; and PMATE is general. As an editor PMATE is basically very satisfactory. It is not, however, a complete word processor since it does not format documents. PMATE is, for the most part, a tool for programmers, who are willing to build macros to provide missing features. The design philosophy behind PMATE was to provide the capability to build word processing functions, rather than the functions themselves.

The features that PMATE has, it uses very effectively. However, there are some important features that PMATE lacks.

PMATE has no on-line help function. The commands used by any text editor are arcane, easily forgotten. Most text editors make provision for user's weak memories by allowing the user to easily find the forgotten command through a 'help'' key. Since PMATE's documentation is weak (more on that later), the help function is sorely missed.

PMATE has no ability to allow the user to access a DOS command or to suspend an edit session, exit to DOS and resume an edit session. While this feature has only recently become common, it would be a nice addition.

There is no ability to search for and selectively change a character string. There is, of course, a global change command, but one would like to be able to search for a string, and decide at that time whether it should be changed or bypassed. Again, one could program a macro to do this, by using one function key to do the search and a different function key to do the change, but it would be nice to have that feature already in place.

PMATE allows deletion of characters, using the Del key; words, using Ctrl W, and portions of lines using Ctrl End. It allows block moves by defining the beginning and end of a block with commands. However, there is no provision to 'delete a sentence'' or ''move a paragraph'' as defined by English punctuation and carriage returns. Since we need to move and delete sentences and paragraphs, as well as of characters and words, the functions of defining and moving blocks of text is less friendly than it could be.

The shortcomings and omissions mentioned above are frustrating in an editor that could have been a contender. However, there are more serious deficiencies.

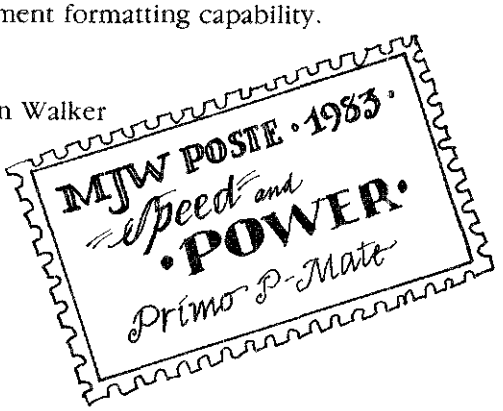
PMATE does not trap disk errors as thoroughly as would be wished. An error such as having write protection on a target diskette causes PMATE to terminate and discard all your work. A ''disk full'' condition, occurring at the end of a marathon session preparing materials for an important seminar, was a bit hair-raising. PMATE recovered itself and did not lose the file being edited; however, the target diskette's file allocation table was ruined. There were three other important files on that diskette which were unrecoverable by normal means. Fortunately, we had some excellent utilities written by Peter Norton, and saved the files through an hour's careful work with UNERASE. Now we're more careful to back up to a second diskette.

Documentation is part of any software product, and ought to reflect the care and professionalism with which the entire product is built. PMATE's documentation reflects very poorly on the software authors. Phoenix Software Associates wrote PMATE and commissioned the documentation from Michael Aronson of Aox Inc. The manual we received was not up to the standards we've come to expect. That edition was for 8-bit CP/M machines and their concepts and usages.

Our main disappointment is that PMATE is not a finished product, and apparently never will be, since Lifeboat (or perhaps Phoenix) seems not to be committed to further development. Very little marketing of PMATE has been done (at least on the West coast) and that makes it hard to find other PMATE users, with whom one might exchange macros and helpful hints. Perhaps a COMPUSERVE SIG would provide contact with a community of users.

In summary, we find that PMATE, as a text editor, provides exciting speed and power, certainly going beyond any IBM mainframe text editor. It's very suitable for preparation of program source code, which has minimal document preparation requirements. As a document processor, it has unrealized potential. We wish the authors would complete their fine initial effort, clean up the shortcomings and publish the macros which would add document formatting capability.

Marilyn and John Walker
Alameda, CA



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Marketing Your Own Software

pj interview

MARKETING YOUR OWN SOFTWARE

an interview with Brett Salter,
the author of PEEKS & POKES

PJ: When did you first start marketing your own software?

BRETT: A little over two years ago I decided to get into business for myself, doing mainframe data base consulting. My background had been with Intel in California. I moved back to Atlanta and then the IBM PC came along.

I got a PC and spent some time working with it. Living off of money I had accumulated from consulting. I finished a couple of products which were then marketed by another company. The results of that were disappointing so I decided to go ahead and sell any new products myself.

PJ: What were your first products?

BRETT: SCREENMASTER, which is a full screen handler for BASIC programs. I use it a lot whenever I do work in BASIC. (As soon as time allows I will begin using the C language. There is a copy of Microsofts C compiler sitting on my desk waiting patiently for some attention). Another product was COMPAREMASTER, we got stuck in the MASTER series of names to try and compete with the VisiCorp's style of brand name marketing. A text processor (formatter) was one other project I completed just prior to PEEKS & POKES.

The text formatter was involved, and as the saying goes, if you ever want a challenge, try and write a text editor. There are so many combinations and permutations to worry about. So many events are happening at the same time that it becomes enormously complex.

PJ: How did you get the idea for PEEKS & POKES?

BRETT: Well, I had put together a list of peeks and pokes and given it to a number of people. It was very well received. Consistent comments along the lines of "this is the type of thing people need to know." Finally I said to myself — hay maybe there is some truth to that. It was important to me to have it be more than just a list of subroutines, so I added some assembly language for BIOS and function calls. Then in November of last year I went out to COMDEX in Las Vegas to peddle it around and watch various reactions to it. The response was enthusiastic so I went home and poured on the coal to make it as complete as possible. Polished it up a bit. (My wife insisted I keep a scrap book of my development work and after looking back on my initial list of features I can only laugh. It ended up nothing like my first concept of the whole thing. It evolved drastically during this period of time.) Last spring PEEKS & POKES was finally released.

My new product INSIDE TRACK (previously called PC POWER) is a continuation of the same style of development. I just start out with a few routines and then as something comes along that looks good I try and stuff it in there. "All the software that fits" could be my byline.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ: Why not start calling them "Salter's Source Code" to give yourself more name recognition?

BRETT: That's true, you have a point. What I need to do is write several articles for various magazines to bring more exposure to who has written the products. It just requires putting down my coding pencil and making the time to put together some articles.

PJ: Too many people seem to miss the tremendous value a set of utility routines can offer. The savings in development costs pays for them many times over.

BRETT: One of my dealers told me that name recognition is definitely a problem. He said that everybody who has PEEKS really loves it. An approach I am working on now to help the name issue is to try and tie all the products together under one logo. The word "Master" has been so over used that it is no longer a contender. Obviously a word like "Visi" carried along with each product is a prominent selling point.

PJ: But why not use your own name?

BRETT: Believe it or not I had trouble putting my name on the front cover of PEEKS & POKES. Again that was my wife who insisted that it is very much like a book so the authors name should go on the front cover. I said OK I'll do it. It is important for me not too feel to egotistical for the market place.

What I will try and do is build a reputation using the underground path. Create a good product at a reasonable price and build upward from there. It is more my style. The main thing is to offer something that the market wants. If you succeed then the next problem becomes completing a second package. What does LOTUS do for a follow up to 1-2-3?

PJ: In your own marketing, what worked and what didn't?

BRETT: I tried a lot of different methods with PEEKS & POKES. Dealing with users groups was a horrible failure. I wrote a letter to each one offering a discount. I even went so far as to include a photo and press release. I received maybe one or two responses from all the clubs. The other big disappointment was trying the mail order houses. I sent out over 70 of these and got one response. One dealer in Texas responded.

I had supplied a sheet of paper explaining how to unprotect BASIC programs, and got a 24% order rate from them when I sent out a flyer. This reinforced my idea about supplying a product to a group that knows you already. They trust that you have something which will benefit them again.

What else . . . . I tried some ads in the larger users group newsletters. Pretty disappointing. I tracked each ad to see what worked. The Boston users group was the best, but still a small return. Ads in magazines have worked well, especially Softalk. Programmer's Journal has worked well in comparison to its circulation. One thing that has been extremely effective has been press releases. You hire a good photographer to take a quality picture of your package and then send it with a two page description of the features to the "New Products" departments. Double space the press copy on your letter head with no more that two pages of description. Byte gave me a half page in their new products section. If I had had to pay for that ad it would have cost me several thousand dollars. A good photograph really helped. BYTE brought in over a 1,000 inquiries, the bingo card variety. Better be ready for them.

PJ: Did you try computer store dealers?

BRETT: I tried a round of letters to dealers. I picked on the states that were biggest for me with the unprotector sheet. They were California, Texas, Illinois, and Canada.

PJ: Canada has a very intense interest in software. Both in producing quality products and in purchasing programmer related packages.

BRETT: I had heard that about Canada and was very well received there. After the initial flyer compaign I still need to do another one to Pennsylvania, New York, Florida, Ohio, etc. I sent ad material to the dealers using the list from that one issue of PC listing everyone.

The photograph is usually included to show that the product exists and is ready to ship. Many people are very fast on the trigger and want it immediately. Too many companies advertise 6 months before the product is ready. Consequently we all have to fight the attitude of "oh I see the ad, probably it will be ready in 3 or 4 months." It is a personal point of pride with me to be able to ship before the advertising begins. This has its drawbacks because the magazine lead time is 3 months minimum.

PJ: How was your first Faire?

BRETT: The one I went to in San Francisco was just great. We sold everything we took including demo copies.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ: That is unusual compared to what I have heard elsewhere. Many companies have booths to establish their financial stance and especially to make contacts, either with dealers or investors.

BRETT: It was very rewarding just to talk to everyone that stopped by the booth. There was only one conflict (or eventual savings as it were) that occurred at the faire. I discovered that the name I had picked for my second product had already been used by another prominent firm. This resulted in a quick change to the current name of INSIDE TRACK to avoid a copyright infringement.

PJ: How did you resolve the name conflict?

BRETT: We had to do a name search in Washington D.C. Your lawyer will contact the appropriate person in Washington who will do the search. They did the trade mark search in one day.

PJ: They searched for the name INSIDE TRACK?

BRETT: Yes. The cost for the Washington part is about $75. And then there is about a $25 charge to write up the summary report and recommendations. As I go ahead and market additional volumes of INSIDE TRACK I will go for a complete trade mark registration.

PJ: What has been the hardest part of doing your own marketing?

BRETT: The many details of publishing and promoting your product. The Post Office is my continuing biggest pain. Also finding appropriate shipping packaging and binder suppliers took a long time. You could write a long article about this aspect alone. The leg work takes a lot of R&D time that could be better spent elsewhere. Pinning down the printing costs is a big hassle. In the mail order business you spend a lot on both postage and printing. It is not obvious at first exactly how to go about it.

The local university offered a course on direct mailing and it was worth every cent I spent on it. Two fellows taught it that sell knifes through the mail. They presented a lot of tips that I would have otherwise had to learn on my own.

TELECOMMUNICATIONS TYPESETTING

COMPUTER OWNERS: If you have a modem, text editor, and communications program, you should consider the improved readability and cost savings provided by *TYPESETTING* your program documentation, manuscript, newsletter, or other text instead of just reproducing it from line printer or daisy-wheel output. Computer typesetting offers you high quality, space-saving copy that creates the professional image you want! Hundreds of type styles to choose from with 8 styles and 12 sizes "on line." And it's easy to encode your copy with our designated typesetting commands.

COMPLETE CONFIDENTIALITY GUARANTEED
— bonded for your protection —

PUBLICATION DESIGN, EDITING, & PRODUCTION

Editing & Design Services Inc.

30 East 13th Avenue Eugene, Oregon 97401

Call Collect 503/683-2657

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ: What were a few that they mentioned?

BRETT: Their main byline was that mail order business is a carnival, so do whatever you have to do to get someones attention. There is a lot of truth to that. Another tip was to call a shipping charge just "shipping" and not "shipping and handling" . Most people feel that they have already paid for the handling. The other is to make it easy to buy, meaning use Master Card and Visa. It is amazing how many ads you see that do not give you enough information to buy the product immediately. I have priced PEEKS & POKES low enough so that is is easy to buy.

PJ: What has been your percentage of charged orders?

BRETT: Probably overall about 30%. Checks tend to come through the mail and charge cards over the phone.

PJ: Do you feel that you have a good handle on the self marketing approach?

BRETT: Reasonably so, but I am still learning. I feel I need a better concept for my display advertising. I have two other companies that are marketing PEEKS & POKES. Also Prentice-Hall has expressed an interest.

I have two major problems with establishing myself in the retail market. One is the price. A dealer does not want to carry a product for $30. They desire the profit margin that they get with a $100 package. Just look at Bruce & James WordVision. They are selling it for $50. A dealer will have to spend the same amount of time selling each copy of WordVision that he spends selling a copy of WordStar. The reluctance on their part is obvious.

The other kind of resistance I have found is that dealers are being overwhelmed with independent offers from software companies. I had one dealer here locally who told me to get in with one of the major software distributors like Sofsel. The dealer said they were getting bombarded with letters from all over the country about new packages. They do not wish to deal with a lot of independent suppliers for their software, they would much rather deal with one or two.

I think that Bruce & James have the right idea about low priced software but it may take a while. Prices are definitely dropping for software in general. The best place for PEEKS & POKES so far has been the software only stores. Most normal computer stores tend to be more hardware oriented because of the larger profits.

All in all my personal foray into software marketing has been a great deal of fun. The real joy is finally assembling that first copy of a brand new product. □

INSIDE TRACK is a new product very much along the lines of PEEKS & POKES only more extensive. A few of the source code features it contains are: copying memory from one location to another, accessing the Program Segment Prefix to get DOS control from your program, dynamically scheduling up to four programs to run in a batch file from within a program, execute any other program while keeping yours resident, read and modify the diskette parameter table, controlling rebooting your system, etc. It is available for $45 from: Data Base Decisions 14 Bonnie Lane Atlanta, GA 30328 ph. 404-256-3860

SafeCopy backs up all IBM PC Software.

There are no exceptions. Copy-protected software is copied readily. **SafeCopy** needs no complicated parameters.

Requirements:

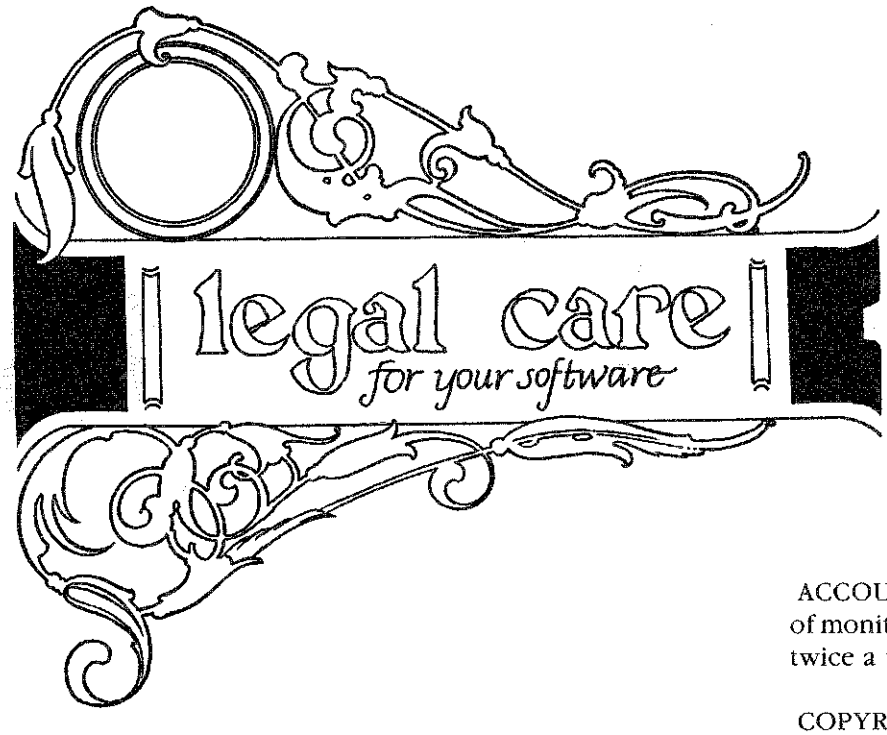
IBM Personal Computer or XT.
48k bytes of memory.
one diskette drive.
SafeCopy will run faster with more memory or another drive.

SafeCopy is revised monthly, to keep up with the latest in copy protection. You may get a new edition at any time for a $12 trade in fee.

SafeCopy is available at a price of $50 US funds from:

Quaid Software Products
620 Jarvis Street, Suite 2412
Toronto, Ontario, Canada M4Y 2R8
Telephone (416) 961-8243

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



AN AGREEMENT OUTLINE

by Robert W. Dunaway of Remer, Remer & Dunaway

Here is a checklist outline of a license agreement with a few additional comments. This checklist can be used by the programmer when drafting or reviewing a license agreement. The checklist was taken from "Legal Care For Your Software" by Dan Remer.

There are some problem areas in license agreements which I will mention, based upon the outline included from the book.

INTRODUCTION - The parties to an agreement should be identified correctly. The introduction should state the business status of the parties. People doing business under a fictitious, partnership or corporate name will be signing their own name on behalf of the business.

LICENSE - The license can be exclusive of non-exclusive, worldwide or limited geographically, etc. Many licenses are restricted to certain machines. The license granted should be very clear as to the scope granted to the publisher.

ROYALTIES - The programmer can be paid in several different ways. The most common is the percentage royalty per sale of the program. Another way is to negotiate a onetime lump sum payment for use of the program. This is called a "paid up" license. The comments on the checklist are very helpful. Royalty advances and minimums are important items for negotiation.

ACCOUNTING - The programmer must have some way of monitoring sales of the program. This is typically done twice a year for accuracy.

COPYRIGHTS - The programmer may want to keep the program copyrighted in his own name. This is especially true in non-exclusive licensing arrangements.

TERMINATION - It is very important for the programmer to be able to get the program back if the publisher does a bad job of marketing. This can be done through minimum royalty payments that must be made by the publisher (or the program reverts back to the programmer). It can also be done in the contract termination section of an agreement. For example, the agreement could provide that the programmer can terminate the license agreement upon 30 days notice after the agreement has been in effect one year.

OUTLINE OF A LICENSE AGREEMENT

USE THIS OUTLINE AS A CHECKLIST FOR ASSEMBLING YOUR OWN LICENSE AGREEMENT.

SECTION I—INTRODUCTION

A. Who are the parties?

B. What is the program?

C. What type of agreement is it? (e.g., license)

SECTION II—DEFINITIONS

A. Terms with different or ambiguous meanings are defined.

SECTION III—ITEMS PROVIDED BY DEVELOPER

A. What exactly is the program that the Developer is licensing to the Publisher?

B. In what form is the program to be delivered?
   1. Does the Publisher get a copy of the source code?
   2. If so, does Publisher set full source-code documentation?

C. Does Developer also furnish a user manual?

D. What does the Publisher not get?

SECTION IV—DELIVERY SCHEDULE

A. When are the item(s) from Section III due to be delivered to Publisher?

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SECTION V—PROGRAM MAINTENANCE, MODIFICATION, AND TRAINING

A. What, if any, are the Developer's responsibilities once the Publisher takes over?

B. Who pays for fixing bugs and any program recalls?

C. Does Developer have to provide program updates?
   1. If so, does Developer get paid for them?
   2. What if the Developer can't or won't provide updates?

SECTION VI—THE LICENSE

A. What exactly is the *subject* of the license?

B. What is the *duration* of the license?

C. What is the geographical and market scope of the license?

SECTION VII—ACCEPTANCE

A. What must the Publisher do to show that the program has been accepted?

B. How much time does the Publisher get to evaluate and test the program prior to acceptance?

SECTION VIII—ROYALTIES

A. What royalty percentage is to be paid the Developer on each program?
   1. Is there a minimum dollar amount per sale?

B. How are royalties calculated?

C. Is there a minimum guaranteed royalty even if the program doesn't sell well?

D. Is there an advance?
   1. If so, how much?
   2. Is it refundable if the program doesn't sell?

E. When are royalties to be paid?
   1. When do they become owing (accrue)?

F. How many free copies does Publisher get?

G. How many free copies does Programmer get for personal use?

H. Are royalties to be paid on the manual?
   1. How much if the manual is sold with program?
   2. How much if it is sold separately?

I. Are royalties to be paid on dealer demos?

SECTION IX—ACCOUNTING

A. What records must Publisher keep and what access does Developer have to them?

SECTION X—WARRANTIES

A. What does the Developer warrant?

SECTION XI—INDEMNIFICATION

A. Under what conditions will Developer indemnify Publisher?
   1. For suits regarding copyright or trade secret?
   2. For bugs or recalls?

B. Under what conditions will Publisher indemnify Developer?
   1. Law suits concerning the quality of the program?

C. What are the dollar limits to the above indemnifications?

SECTION XII—COPYRIGHTS

A. Who is responsible for copyrighting the program?

SECTION XIII—CONTRACT TERMINATION

A. Who can terminate the contract?
   1. Why can this be done?
   2. How can this be done?

B. What, if any, contract provisions will survive termination?

SECTION XIV—ARBITRATION

A. Do the parties want binding arbitration to govern contract disputes?

SECTION XV—SOURCE-CODE ESCROW

SECTION XVI—GENERAL

A. Everything that was not included above.

B. What happens if either party dies or goes under?

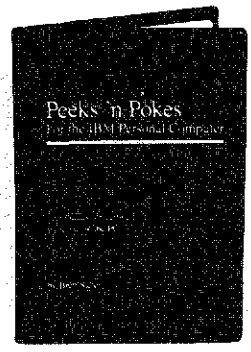
ATTACHMENTS

1. Program description.
2. Delivery timetable (if too complex to fit in main body of contract).
3. Manual description (if too complex to fit in main body of contract).
4. Royalty scales (if too complex to fit in main body of contract).

(C) COPYRIGHT BY DANIEL REMER. ALL RIGHTS RESERVED

Price $19.95
Available from:
NOLO PRESS
950 Parker Street
Berkeley, CA 94710

Peeks'n Pokes For the IBM PC puts you in control of your PC!



Are you writing programs in BASIC or Pascal? This unique product has a disk with 58 programs and a 38-page manual that helps you get 'underneath the covers' of the PC. Learn how to use PEEK, POKE, INP, OUT, and DOS/BIOS function calls to do what you want, fast! Do you want to perform functions not available from BASIC or Pascal? It's all explained in the manual and demonstrated in the sample programs. You can increase the capabilities of your programs while making them even easier to use.

The easy to learn techniques save you time and increase your productivity.

Peeks 'n Pokes shows you how to:

- Access the system's configuration
- Unprotect BASIC programs
- Scroll part or all of the screen
- Access the file directory
- Logically swap printers
- Read and change the keyboard
- Find more Peeks 'n Pokes
- And much more...

All this for only $30.00 plus $2.50 shipping. MasterCard and VISA accepted.

Data Base Decisions • 14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE COPY PROTECTION WAR

by Robert T. McQuaid

There is a battle going on between the software suppliers who want to keep you from making copies of your software, and the copy utility companies who try to copy everything in the marketplace. Each time a new copy protection scheme appears, the authors of copy utilities go to work to create a product that copies it. Each time a software supplier wants to copy protect his software, he buys the latest copy utility, and creates something it can not copy. This is an inside article written by one of the participants in the war. I am the author of a utility that copies all software diskettes for the IBM Personal Computer. If you think that copy protection is a science, this article will correct your illusions.

Early Copy Protection:

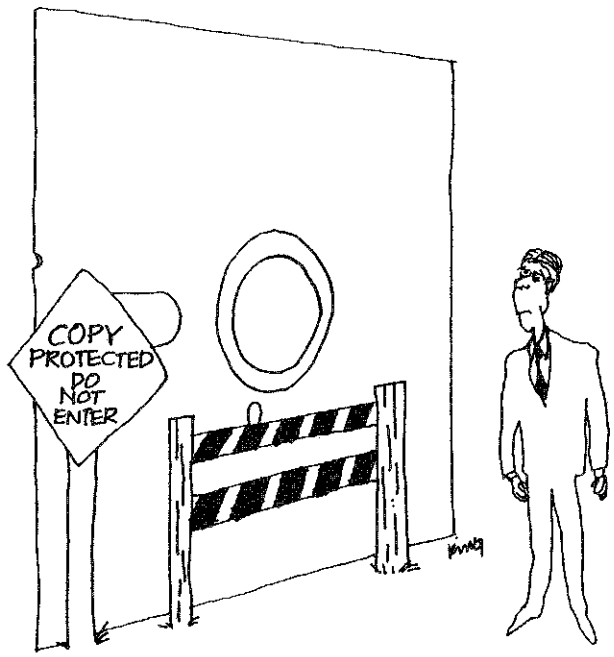
When the IBM Personal Computer appeared in 1981, it came with a copy utility, Diskcopy, as part of the operating system. This utility copied diskettes conforming to the IBM conventions. IBM itself distributed some copy protected software with the original system, and the copy protection was achieved by using the id information stored on the diskettes. When a diskette is formatted, a packet of information called the id field goes in front of each data sector. A sector number goes in the id field. Normally, IBM used only sectors 1 to 8, but the copy protected diskettes used a sector number above that range, causing Diskcopy to stop at the bad sectors. Copying such a diskette was a nuisance, but simple. You used the service calls in the IBM operating system and copied all good information from one diskette to the other. You could ignore the bad sectors, because the copied program did not use them in any way.

It soon became obvious that this was not very secure. A simple improvement was to put some of the information required by the software in the bad sectors. Now, a copy utility that bypassed the bad sectors would omit some vital information, and the copy would be useless. This also proved quite easy to circumvent, since the floppy disk controller in the IBM Personal Computer has the ability to read the id fields, giving a report on their contents. It is a simple matter to write a utility that reads all of the id fields, and formats a new diskette with the same id fields as the old one.

This led to the development of a copy protection scheme that could not be copied by reading the id fields and duplicating them on the new diskette. The id field contains, in addition to a sector number, a length. The copy protector puts five sectors on a track, say, 1-2-3-4-5. Sector 2 has an unusually long length code, so writing to sector 2 erases the id for sector 3. When the diskette is distributed, it has only sectors 1-2-4-5, and this is what the copy utility transcribes to the duplicate. But when it writes to sector 2, it erases sector 4, leaving only 1-2-5. The software on the copy protected diskette checks for the presence of sector 4 and stops when it is missing. Defeating this technique requires careful use of a timer to measure the angular position of each id on the diskette.

Current Copy Protection:

That was the state of the art when the IBM Personal Computer was introduced. Several clever ideas have appeared since. One idea was to use the CRC, the cyclic redundancy check, recorded at the end of each data field on a diskette. This CRC is computed from the data during recording, then put on the diskette after the data itself. While reading from the diskette, the CRC is computed again and compared with the recorded value. When the two values agree, the data is presumed to be correct, but when they differ the disk controller



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

complains. The new idea was to record a deliberate error in the CRC. Any copy program that tried to read the diskette would abort when it encountered the errors. Of course, some one could write a utility that ignored those errors, and accepted the data supplied by the controller, but the scheme included a devious twist. When the copied software got in control, it read the diskette, and insisted on a bad CRC. If it saw a good CRC, it treated the diskette as invalid and refused to run. There is no legitimate way to create a bad CRC on the IBM Personal Computer. This stops amateur copiers, but most copy utilities can duplicate this kind of protection.

Another idea was to put several identical id fields on a track. Until you figure out what is happening, this produces hopeless confusion. This scheme is used by only a few companies, and it seems that few utilities have bothered to include the ability to duplicate it.

One simple but effective idea was to leave a blank track on the diskette. The inventors of this scheme know how the IBM Personal Computer can format a track with data, but do not know how to erase a track completely. Copy utilities know how to do so.

A big leap forward occurred with the discovery that the disk controller of the IBM Personal Computer responds in an unusual way to a bad CRC in the id field. Earlier, I omitted the detail, but each sector has an id field and a data field, each with its own CRC. A bad CRC in the id field creates a sector that you can detect if you know its number, but not otherwise. That is, if the id contains an R field of 203 and you try to read sector 203, you get an error code indicating that sector 203 is on the diskette in erroneous form. But when you try to use the read id function of the controller, you get no indication that the sector exists. This works something like a combination lock. When you know the combination, you can open the lock quickly, but when you do not know it, all you can do is try lots of possibilities. The lock will give you no clue about what the right combination is. This scheme is breakable, but only with advanced techniques.

Recently a product appeared that had several sectors with proper id fields, but no data fields. The authors of this scheme no doubt told their superiors that there was no way to duplicate this format using the disk controller of the IBM Personal Computer. They thought they were safe. They were not.

The Future of Copy Protection:

The Apple is a more primitive, but more mature system than the IBM Personal Computer. The same war has been going on with the Apple, but it has reached a more advanced state. The result of the continuing battle has been the emergence of more and more bizarre copy protection schemes. A similar future may be in store for the IBM PC.

There are several follies indulged in by creators of copy protected diskettes. One is the idea that they can get away with making things difficult for the user. A few packages, such as Volkswriter, have abandoned copy protection to make life easier for legitimate purchasers. For some kinds of products, notably games, it seems that copy protection will always be necessary. For these, there is another kind of folly. Most of the copy protection schemes were designed in a few weeks by programmers with no previous experience in copy protection. Their main tool is making things complicated. They think that their objective is to design something that will stop 99% of their purchasers from copying their diskettes, and they succeed. They fail to consider that their adversary is not the rank and file user, but the specialist who has seen all of the copy protection techniques, and knows a lot of tricks with the disk controller. Each new invention, when analyzed, only enhances the repetoire of the utility specialist.

Sad Ending:

In the course of investigating copy protection, I have considered who will win the battle in the end. I regret to say that the laws of nature favor those who want to inhibit copying of their products. I have found techniques that are much more secure than anything now in use, and can be made absolutely secure against general purpose copy utilities. The only way to copy such a diskette would be by a custom analysis of each diskette to be copied. Now the good news. The makers of copy protected software are not purchasing their methods from copy utility experts, so you have several more years ahead of you in which you may back up your copy protected diskettes. □

The author, Robert T. McQuaid, is known to the outside world as a computer consultant, but to the inside world as a senior hacker. Among his products for the IBM Personal Computer are SafeCopy, a utility that duplicates copy protected diskettes, and Disk Explorer, a utility that allows examination and patching of copy protected diskettes.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MULTITASKING

MULTI-TASKING

by Frankie Detesto

There are a number of multi-tasking products out for the PC. Three that we have looked at are MEMORY-SHIFT, MULTI-JOB, and MULTI-LINK. They each have their distinct high points.

MEMORY-SHIFT ($99)
NABS
642 Office Parkway
St. Louis, MO 63141
ph. 800-325-1485

With MEMORY-SHIFT you can have up to 10 partitions running. You allocate them when you boot the system. Each partition must have DOS booted separately. You may run whatever program you wish in each partition, but some fancy ones will not work, like LOTUS 1-2-3 and a few others that have copy protection and play around with the DOS internals.

You can therefore have your favorite word processor in partition 1 and a dBASEII program in partition 2. You worry a little about which disk drive holds the right files. It is hard to avoid swapping them around, because if you did then you will have a really FAT problem.

When you leave a partition it is DEAD. This avoided a lot of concurrent problems, and you cannot call MEMORY-SHIFT a multi-tasking product because of this. It is a multipartition product. Only the job you are in is active. It is quite exciting to switch among partitions, and I do not know what I would do with more than 2 or 3 even though you are allowed 10.

Problems abound. The biggest one is hanging the system. This happens quite frequently unless you are very careful. Perhaps your disk drive turns on and won't switch off or the key board is dead. You can also forget about using your RAMDISK. This is somehow incompatable with MEMORY-SHIFT, or at least magnifies the hang up problems.

There is one feature of MEMORY-SHIFT that is absolutely worth the $99 price tag. Let's say you are in WordStar in partition 1 and you have a VisiCalc screen in partition 2. You have finally figured out exactly the budget you would like using VisiCalc and would like to transfer it to your word processor. MEMORY-SHIFT allows you to SAVE the VisiCalc screen to a storage area, switch partition to WordStar, and then haul the VisiCalc screen back out into your text file. No conversions, tuples, and other Mickey Mouse stuff. I would keep MEMORY-SHIFT around with all its hangups and use it once or twice a month just to transfer important screens into my word processor.

MULTI-LINK
The Software Link, Inc
6700 23-B Roswell Road
Atlanta, GA 30328
ph. 404-255-1254

The distinct feature of MULTI-LINK is the fact that it is designed to emulate dumb terminals in its background partitions. You will find yourself setting up the other partitions just like adjusting normal terminal settings, like baud rate, ESC codes, etc.

Several terminal types are covered; ADDS, ADM, Televideo, DEC, IBM, Zenith. Essentially you are setting up an emulator that will translate the necessary codes to control end-of-line, clear to bottom of screen, etc.

You can easily set up the modem and run a communications program in partition 2. Then someone in Idaho can be using your PC in the background while you are using it in the foreground. A very intriguing thought.

The caveats are the fact that you are in the old dumb terminal mode if you are alone on the system and you are in the background. This becomes a drawback when you are using your memory mapped word processor that does not function properly using a dumb terminal screen control. After being used to so many dedicated and fine tuned products for a single user, you will have to learn to do without them in order to use MULTI-LINK. But the the primary scope of the product is to provide remote or serial port CRT access to your IBM, and that it does very well.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MULTI-JOB
B&L Computer COnsultants
7337 Northview, Suite B
Boise, ID 83704
ph. 208-377-8088

The ability to have concurrent processing is a main focus of MULTI-JOB. You can really get a handle on partition size and job priorities. If you are doing scientific research that requires long calculation runs then it would be a tremendous help. Assigning job priorities is quite necessary in a mathematical analysis setting. The manual mentions that MULTI-JOB works especially well with the Macro Assembler. You can start a batch assembly series in one partition and then go back to editing in another partition.

Numerous memory PATCHes are mentioned in the manual. Some are necessary to disable output to a port being used under BASIC when another job is trying to use the COM port for telecommunications.

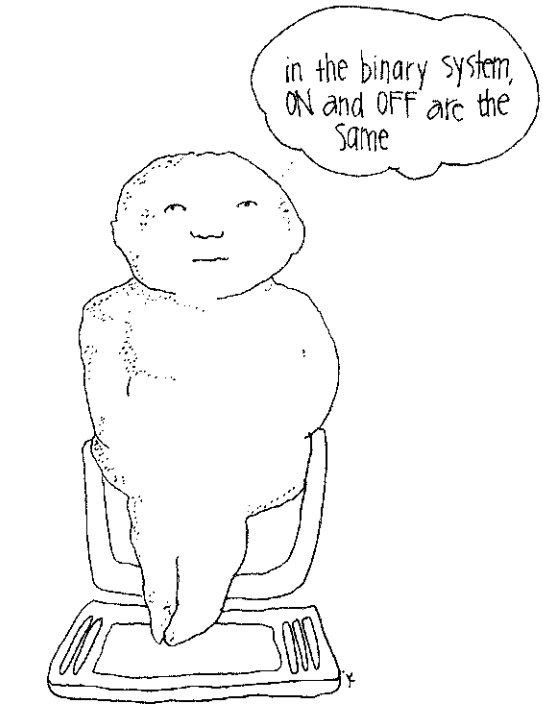
You can have a number of AUTOEXEC.BAT files under MULTI-LINK so that you can boot up all the partitions at once. This is a definite plus. Only four jobs are allowed which should be plenty.

SUMMARY

Each of these products have their own particular thrust. MEMORY-SHIFT can transfer a screen of data between partitions very easily, MULTI-LINK is expressly designed for attaching CRT's to your PC, and MULTI-JOB is a concurrent system that is tailored for multiple job processing. They have begun the adventure of multi-tasking on the IBM PC. As leaders, they may be forgiven the rough edges of their products.

It is time that we had some standards for multi processing. As each new sophisticated product hits the streets the 'stepping on each others toes' becomes more of a forgone conclusion rather than an infrequent side effect. The 3 Bears adage of "someone is sleeping in my Alt-F9 bed" is now a reality.

Will we have to give up memory mapped displays so we can have concurrency? Can't we set up a key definition protocol so our programs will check for previous redefinitions? How about daisy chaining them together? If a MULTI allocation table were present we could all tell which partition we were in, which one was active, who has what devices and so on. Do we have to leave this up to Microsoft or DRI? We could establish one ourselves. The first time you try to use PROKEY, your RAMDISK, LOTUS 1-2-3, a MULTI partition, and a dedicated wordprocessor at the same time, the writing on the walls becomes especially clear. We are waist deep and just sinking deeper. □



ZEN PROGRAMMER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PASCAL CHECK PROGRAM

by Corey Morrell

Here's a short PASCAL program that will demonstrate some of the basics of PASCAL. There are far more bells and whistles in the PASCAL language than are contained in this program, but I tried to keep it straight forward.

This program can be used to balance your checkbook at the end of each month. It asks for your beginning balance, the amount of each deposit, the amount of each check written, and the amounts of any service charges. It then displays the beginning balance, the total amount of deposits, the total amount of checks written, the total amount of service charges, and the new balance.

Some things to note before you continue:

The line numbers at the left are not PASCAL code. I added them after testing this program, and they are for annotation purposes only. The program would not compile if they were present.

The program consists of four procedures and the main program portion. PASCAL requires that you define all procedures before they are called, hence the main program portion starts after all the procedures are defined. When this program executes, you can consider it starting at line 86.

The name of a procedure in a line is enough to call it. Lines 95, 96, 97, and 98 are calls to procedures. They are equivalent to GOSUB's in BASIC.

There are no GOTO's in this program. It is an example of 'Structured Code" , or "Top-Down" programming. By reading the main program portion at the bottom, you can get an idea of the logic without jumping all over the place.

The semicolon (;) is used as a statement delimitter in PASCAL. It is required after all statements except BEGIN, or those followed by an END or ELSE.

The indenting of the code is not required by PASCAL. I included it for readability purposes only.

After the code itself is a blow-by-blow commentary on the source using the line numbers for reference. □

A SAMPLE RUN

```
A>CHKBAL

ENTER BEGINNING BALANCE : 300.00


ENTER 1 DEPOSIT AMT TO EACH D?
ENTER  0  WHEN FINISHED

D? 150.00
D? 200.00
D? 475.00
D? 0


ENTER 1 CHECK AMT TO EACH  C?
ENTER  0  WHEN FINISHED

C? 23.50
C? 45.00
C? 36.43
C? 23.89
C? 123.65
C? 89.76
C? 0


ENTER 1 SRV CHG AMT TO EACH  S?
ENTER  0  WHEN FINISHED

S? 4.50
S? 0

------------------------------------------------

BEGINNING BALANCE      =    300.00

TOTAL DEPOSITS MADE    =    825.00

TOTAL CHECKS WRITTEN   =    342.23

TOTAL SERVICE CHARGES  =    4.50

          NEW BALANCE  =    778.27
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
1    PROGRAM CHKBAL(INPUT,OUTPUT);
2    VAR DEPOSIT,DEPTOT,CHECK,CHECKTOT:REAL;
3        SERVCHG,SRVCHTOT,BALANCE,NEWBAL:REAL;
4    (*                                          *)
5    PROCEDURE DEPOSITS;
6    VAR DONE:BOOLEAN;
7    BEGIN
8      DONE:=FALSE;
9      WRITELN;
10     WRITELN;
11     WRITELN('ENTER 1 DEPOSIT AMT TO EACH D?');
12     WRITELN('ENTER  0  WHEN FINISHED');
13     WRITELN;
14     WHILE NOT DONE DO
15       BEGIN
16         WRITE('D? ');
17         READLN(DEPOSIT);
18         IF DEPOSIT=0 THEN DONE:=TRUE
19         ELSE
20           BEGIN
21             NEWBAL:=NEWBAL+DEPOSIT;
22             DEPTOT:=DEPTOT+DEPOSIT
23           END (* ELSE *)
24       END (* WHILE NOT DONE *)
25   END; (* PROCEDURE DEPOSITS *)
26   (*                                          *)
27   PROCEDURE CHECKS;
28   VAR DONE:BOOLEAN;
29   BEGIN
30     DONE:=FALSE;
31     WRITELN;
32     WRITELN;
33     WRITELN('ENTER 1 CHECK AMT TO EACH  C?');
34     WRITELN('ENTER  0  WHEN FINISHED');
35     WRITELN;
36     WHILE NOT DONE DO
37       BEGIN
38         WRITE('C? ');
39         READLN(CHECK);
40         IF CHECK=0 THEN DONE:=TRUE
41         ELSE CHECKTOT:=CHECKTOT+CHECK
42       END; (* WHILE NOT DONE *)
43     NEWBAL:=NEWBAL-CHECKTOT
44   END; (* PROCEDURE CHECKS *)
45   (*                                          *)
46   PROCEDURE SCHARGES;
47   VAR DONE:BOOLEAN;
48   BEGIN
49     DONE:=FALSE;
50     WRITELN;
51     WRITELN;
52     WRITELN('ENTER 1 SRV CHG AMT TO EACH  S?');
53     WRITELN('ENTER  0  WHEN FINISHED');
54     WRITELN;
55     WHILE NOT DONE DO
56       BEGIN
57         WRITE('S? ');
58         READLN(SERVCHG);
59         IF SERVCHG=0 THEN DONE:=TRUE
60         ELSE SRVCHTOT:=SRVCHTOT+SERVCHG
61       END; (* WHILE NOT DONE *)
62     NEWBAL:=NEWBAL-SRVCHTOT
63   END; (* PROCEDURE SCHARGES *)
64   (*                                          *)
65   PROCEDURE FINISHUP;
66   BEGIN
67     WRITELN;
68     WRITELN;
69     WRITELN('-------------------------------------');
70     WRITELN;
71     WRITELN('BEGINNING BALANCE     = ',BALANCE:8:2);
72     WRITELN;
73     WRITELN('TOTAL DEPOSITS MADE   = ',DEPTOT:8:2);
74     WRITELN;
75     WRITELN('TOTAL CHECKS WRITTEN  = ',CHECKTOT:8:2);
76     WRITELN;
77     WRITELN('TOTAL SERVICE CHARGES = ',SRVCHTOT:6:2);
78     WRITELN;
79     WRITELN;
80     WRITELN('          NEW BALANCE  = ',NEWBAL:8:2);
81     WRITELN
82   END; (* PROCEDURE FINISHUP *)
83   (*                                          *)
84   (*            M A I N   P R O G R A M       *)
85   (*                                          *)
86   BEGIN
87     DEPTOT:=0;
88     CHECKTOT:=0;
89     SRVCHTOT:=0;
90     WRITELN;
91     WRITELN;
92     WRITE('ENTER BEGINNING BALANCE : ');
93     READLN(BALANCE);
94     NEWBAL:=BALANCE;
95     DEPOSITS;
96     CHECKS;
97     SCHARGES;
98     FINISHUP;
99     WRITELN;
100    WRITELN;
101    WRITELN('              G O O D   B Y E')
102  END. (* PROGRAM CHKBAL *)
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

1. The program heading. This is required in all PASCAL programs. I chose the name CHKBAL for this one. In parens are the names of any files your program will use. INPUT and OUTPUT are the system defaults for the keyboard and video screen respectively. This is why they are not required in any of the read and write statements.

2-3. The VAR statement declares all those variables as REAL. One can also declare INTEGER and BOOLEAN types, as well as your own user defined types. Since this program declares these variables at the beginning (before any procedures), they are "GLOBAL", that is their scopes include the main program and all the procedures.

4. A comment line with nothing in it used as a blank line. Compare lines 25, 42, 84 etc.. The compiler ignores anything between (* and *).

5-25. The declaration of the procedure DEPOSITS.

5. This names the procedure. Procedures in PASCAL can pass parameters but I haven't used any in this program.

6. Declare the local variable DONE as a BOOLEAN type. BOOLEAN type variables can only take on two values: True or False.

7. Begin the procedure DEPOSITS. All procedures must start with a BEGIN statement and end with an END statement, as must the main program itself (86 and 102).

8. Initialize the variable DONE to False. Assignments in PASCAL are done with the := operator. The = by itself is only used as a relational operator (18 and 40 eg.).

9-10. The WRITELN statement by itself will print a blank line.

11-12. Prints those literals on the screen.

14. The WHILE statement will do everything between its BEGIN and END statements (15-24) repeatedly as long as the condition "NOT DONE" is true. Since we set DONE to False in line 8, "NOT DONE" is true. If the WHILE condition is False at the start, then statements 15-24 will not be executed at all. This as opposed to a REPEAT UNTIL structure or a FOR-NEXT loop in BASIC, each of which will execute their contained statements at least once even if the starting condition is false.

16. The WRITE statement will print what is between the quotes just like the WRITELN statement (11 & 12), but the WRITE will not include a CR-LF pair as will the WRITELN statement. This allows the operator to key in the deposit amount immediately to the right of the D? prompt rather than at the beginning of the next line.

17. Accepts a number with decimal point from the keyboard into the variable DEPOSIT.

18-23. IF-THEN-ELSE structure. It is similar to BASIC's IF-THEN-ELSE, except that in PASCAL you delimit the scope of the statements to be executed for the THEN or the ELSE condition by a BEGIN-END pair; but only if there is more than one statement to be executed. Thus, since the THEN portion only executes one statement (DONE:=TRUE), we stick it on the same line (18) with no BEGIN-END pair. But, since the ELSE portion has two statements to execute (21 & 22), we delimit them with a BEGIN-END pair (20 & 23). Some of you sharper programmers may have noticed that line 21 is wasteful and NEWBAL could have been figured only once outside the loop using DEPTOT. You are correct, and that is how I coded it in the other procedures (40-43 & 59-62). I included line 21 merely to show how one handles multiple statements within the IF-THEN-ELSE structure.

23-25. Each BEGIN statement must be paired with an END statement in PASCAL. PASCAL pairs each END with the closest unpaired BEGIN that it finds above that END. So, 23 & 20, 24 & 15, and 25 & 7 are paired.

27-44. The procedure CHECKS. It inputs all the checks for the month and keeps the total CHECKTOT.

46-63. The procedure SCHARGES. It does the same as CHECKS but for service charges.

71. Prints the literal "Beginning Balance" followed by the value of the variable BALANCE. The ":8:2" mean that the value of BALANCE is to be printed using a total of 8 spaces (including decimal point) with two places to the right of the decimal. It is equivalent in BASIC to: PRINT USING "BEGINNING BALANCE = #####.##";BALANCE

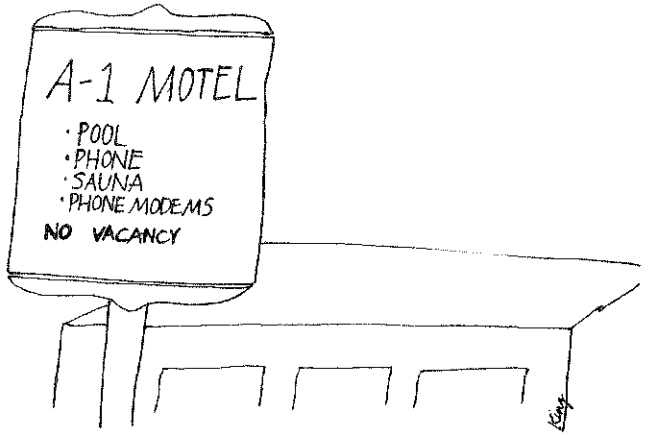
86-102 The main program portion. This is where execution starts.

87-89. Initialize variables.

92-93. Input the beginning balance into BALANCE.

95-98. Call each procedure in turn.

102. The END statement for the program. Notice that it has a period after it. This is required, and is the only place a period is used in PASCAL.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C TRANSPORTABILITY PROBLEMS

by Ken Smolek, author of the C Screen Editor

There was a problem that really got me when I was compiling the same C source code among different C compilers. By definition all arithmetic comparisons of character variables are done as integers. For instance when you pass parameters on the stack if you have a character as a parameter then it gets passed as an integer. It is expanded to 16 bits or whatever form your machine is using. The standard integer size is used. All the numeric calculations are performed with the expanded integers, there is no such thing as a byte add. Your result may be a character when you are done with the addition, but you are adding two integers. That generates a non-obvious problem, which way do you convert it to 16 bits. Do you assume it is an unsigned value, and convert all high bits to zeros or do you assume it is signed and then extend things into the sign bit. Which means that values over 127 will be negative when they are converted. This becomes very significant when you do this type of comparison.

```
char c

if (c = = -1) ... etc
```

Negative one is an integer, and when "c" is expanded it will never match if all extensions are assumed to be unsigned. My solution to this problem among C compilers is to have both sides of a comparison sign extended in whatever way the compiler sees fit. Have a global variable called END for example:

```
char end = -1
```

And then when the comparison is done you end up with two integers converted identically.

```
char c

if (c = = end) ...etc
```

Make sure they are both the same kind of variables. CI-86 has a U option to stop sign extension of characters if you don't want to. Lattice on the other hand just ignores the sign extension. So if you are going from CI-86 to Lattice and you need the sign extension you are in trouble. The program will compile OK but will do very strange things.

DRI's compilers seem to cover signed and unsigned characters. Their compilers do a sign extension unless you call it an unsigned character. But that is not according to K&R or more exactly it is not covered completely. Really, the types defined are only character, integer, unsigned, long, float, and double. CI-86 supports unsigned char, and unsigned long; unsigned float and double are sort of meaningless. It may have been easier it they had defined characters to be sign extended or not. Either way would have been solved the problem. They designed C for easy implementation, so that arithmetic expressions have no specific order of evaluation. For example:

```
(a + b) + (b + c)
```

You have no guaranty that the parser will evaluate the left side before the right side. If you write an expression like the following you are really asking for trouble.

```
(a + b) + ((b = c) + d)
```

Just avoid these at all costs.

One difficulty I am fighting with another machine is that usually a pointer and an integer are the same length. This is implicitly assumed in much of the older UNIX code. On the 68000 machine I am working on a pointer can of course be 32 bits, while an integer is only 16. When you define something as an integer and try and return a pointer in it then you get all sorts of compiler errors.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
char strcpy(s,p)

char *s,*p;

...

return p;
```

Most compilers will catch it, but some will not. Strcpy should be defined as:

```
char *strcpy(),*strcat,..... etc
```

Another variation among compilers is the need for absolutely matching comments. On some a */ will cancel all previous comments like this:

```
/*.....
/*.....
/*.....
*/
```

This is considered balanced, whereas with others you must match them even though they are not consecutive.

```
/*.....*/
/*.....
/*.....*/
*/
```

You can even things out this way. The first version is much nicer to use.

Reads and writes are often different from compiler to compiler. The OPEN command in UNIX does not distinguish between ASCII or binary files whereas CP/M does. Compilers that run on both usually give you options to adjust accordingly. □

BASIC? IF YOU USE IT, YOU NEED BDS!

The BASIC Development System (BDS) is an integrated set of software tools that will greatly increase your programming efficiency and productivity. BDS is **co-resident** with the interpreter to support BASIC's interactive philosophy. You never need to save programs in ASCII and leave BASIC to perform any BDS function. Written entirely in machine language for MAXIMUM SPEED in minimum space, BDS provides the following features:

- **SCROLLING KEYS** Allow you to instantly list the first, last, current, previous, or next program line. Now you can quickly scan through a program to locate the lines you want to modify.
- **SINGLE STEP TRACE** Pauses before executing each new line and waits for keyboard input to continue. The line number is placed in the upper-right corner of the screen so it doesn't interfere with program displays. You may break out of this trace mode at any time to examine variables, then CONTinue tracing. Makes it easy to follow the program logic and catch those hard-to-find bugs.
- **CROSS-REFERENCE** The indispensable programming tool. Provides a sorted list of every BASIC keyword, numeric and string constant, line number, and variable name used in the program and shows where each item is referenced. After a single item is referenced, each line containing the reference can be automatically displayed one at a time for view and/or modification.
- **SUPER RENUM** Gives you all the features that the RENUM command leaves out. Renumber, relocate, or duplicate any block of program lines without affecting other portions of the program.
- **VARIABLE DUMP** A single command will list any or all program variables, along with their current values. Each array element is listed separately. This is a powerful debugging tool, especially when used with SINGLE STEP TRACE.
- **COMPRESS** Reduces the program size to an absolute minimum. Leaves more space for variables and string data and increases execution speed.
- **UNCOMPRESS** Expands a compressed program to make it more readable.

BASIC DEVELOPMENT SYSTEM (BDS).....$79

DCRS is the ULTIMATE research tool for the experienced 8086/8088 Assembly Language programmer. It captures output from the PC-DOS DEBUG utility: prints a disassembly listing marking each referenced address, then a sorted cross-reference of all hexadecimal byte and word values appearing in the disassembly list.

DISASSEMBLY CROSS-REFERENCE SYSTEM (DCRS).....$49

TRY IT! You'll like it, or your MONEY BACK. If you are not satisfied for any reason, return the product within 30 days for a full refund.

TO ORDER NOW, CALL (303) 793-0145

All products require a 64K IBM PC (or XT), one disk drive and PC-DOS 1.10 or 2.00

Add $3.00 for shipping and handling.

SOFTOOL SYSTEMS

8972 E Hampden Ave • Suite 179
Denver, Colorado 80231

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5 WAYS TO COMBINE MODULES

by Art Zachai

The following ASM code is taken from Art Zachai's series of ASM tutorial programs, now available from:

STARWARE
2000 K St. NW
Washington, DC 20006
ph. 202-331-8833

```
;       (C) COPYRIGHT by ART ZACHAI 1983, ALL RIGHTS RESERVED
;
;          5 WAYS TO COMBINE SEPARATE MODULES
;

The IBM Personal Computer MACRO Assembler 02-11-83        PAGE    1-1

; MAIN-1        CONNECTING SEGMENTS AND MODULES

;               METHOD #1  -USING PUBLIC SEGMENT NAMES

;
STACK           SEGMENT STACK   ;"STACK" AS SEGMENT COMBINE TYPE IS MAGIC,
                DW 2 DUP(?)     ; SEE "LINK" IN THE DOS MANUAL.
STACK           ENDS

                EXTRN SUBR1:NEAR

METH_1_CODE     SEGMENT  PUBLIC
                ASSUME CS:METH_1_CODE,SS:STACK

BEGIN_MAIN_1    PROC  FAR               ;SET UP FOR RETURN TO DOS
                PUSH  DS
                XOR   AX,AX
                PUSH  AX
                CALL  SUBR1             ;SUBR1 IS IN THIS SEGMENT, IN ANOTHER MODULE.
                RET                     ;RETURN TO DOS
BEGIN_MAIN_1    ENDP
METH_1_CODE     ENDS
                END BEGIN_MAIN_1

--------

The IBM Personal Computer MACRO Assembler 02-11-83        PAGE    1-1

; SUB1          METHOD 1, MODULE 2

                PUBLIC SUBR1

METH_1_CODE     SEGMENT PUBLIC
                ASSUME  CS:METH_1_CODE
SUBR1           PROC    NEAR
                NOP
                RET
SUBR1           ENDP
METH_1_CODE     ENDS
                END

------

  Start  Stop   Length  Name                    Class
00000H 00011H 0012H  METH_1_CODE
00020H 00023H 0004H  STACK

  Address         Publics by Name

0000:0010       SUBR1

Program entry point at 0000:0000
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; MAIN-2        CONNECTING SEGMENTS AND MODULES

;   METHOD #2  -USING FAR CALLS.

STACK           SEGMENT STACK   ;"STACK" AS SEGMENT COMBINE TYPE IS MAGIC.
                DW 2 DUP(?)
STACK           ENDS

                EXTRN SUBR2:FAR

MAIN_2_CODE     SEGMENT  PUBLIC
                ASSUME CS:MAIN_2_CODE,SS:STACK

BEGIN_MAIN_2    PROC  FAR
                PUSH  DS        ;SET UP FOR RETURN TO DOS
                XOR   AX,AX
                PUSH  AX
                                ;SUBR2 MAY BE IN THIS PHY. SEGMENT,IN ANOTHER
                CALL  SUBR2     ; MODULE, BUT ASSEMBLER CANNOT KNOW WHAT THE
                                ; ASSUME CS: IS IN OTHER MODULES AS IN METHOD 1
                RET             ;RETURN TO DOS
BEGIN_MAIN_2    ENDP
MAIN_2_CODE     ENDS
                END BEGIN_MAIN_2
```

```
   The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; SUB2          METHOD 2, MODULE 2

                PUBLIC SUBR2

SUB2_DATA       SEGMENT PUBLIC
LOCAL_WORD      DW 1234
SUB2_DATA       ENDS

SUB_2_CODE      SEGMENT PUBLIC
                ASSUME CS:SUB_2_CODE, DS:SUB2_DATA
SUBR2           PROC    FAR             ;FAR - CS REFERENCES THIS SEGMENT AT
                MOV     AX,LOCAL_WORD   ; ASSEMBLY TIME,
                JMP     NEXT_INSTR      ; SO A 2-WORD (SEG:OFFSET) CALL
NEXT_INSTR:     NOP                     ; MUST BE GENERATED ELSEWHERE.
                RET                     ;(THE CODE HERE IS NONSENSE, OTHER
SUBR2           ENDP                    ;THAN ILLUSTRATING ADDRESSING)
SUB_2_CODE      ENDS
                END
```

```
 Start  Stop   Length  Name                    Class
 00000H 00009H 000AH  MAIN_2_CODE
 00010H 00013H 0004H  STACK
 00020H 00021H 0002H  SUB2_DATA
 00030H 00037H 0008H  SUB_2_CODE

  Address         Publics by Name

 0003:0000       SUBR2

Program entry point at 0000:0000
```

```
The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; MAIN-3        CONNECTING SEGMENTS AND MODULES

;   METHOD #3  -USING FAR CALLS AND CLASSNAMES. LIKE METHOD #2, EXCEPT SEGMENTS
;     WITH THE SAME CLASSNAME GET CONCATENATED ON PARAGRAPH BOUNDARIES AT LINK
;     TIME. MAKES DEBUGGING EASIER AND PROVIDES A METHOD OF PUTTING DATA IN ONE
;     AREA OF MEMORY, AND CODE IN ANOTHER, PERHAPS EVENTUALLY IN ROM.

STACK           SEGMENT STACK   ;"STACK" AS SEGMENT COMBINE TYPE IS MAGIC.
                DW 2 DUP(?)
STACK           ENDS

                EXTRN   SUBR3:FAR
MAIN_3_CODE     SEGMENT  PUBLIC 'Z_CODE'
                ASSUME CS:MAIN_3_CODE,SS:STACK
BEGIN_MAIN_3    PROC    FAR
                PUSH    DS      ;SET UP FOR RETURN TO DOS
                XOR     AX,AX
                PUSH    AX
                                ;SUBR3 MAY BE IN THIS PHY. SEGMENT,IN ANOTHER
                CALL    SUBR3   ; MODULE, BUT ASSEMBLER CANNOT KNOW WHAT THE
                                ; ASSUME CS: IS IN OTHER MODULES AS IN METHOD 1
                RET             ;RETURN TO DOS
BEGIN_MAIN_3    ENDP
MAIN_3_CODE     ENDS
                END BEGIN_MAIN_3
```

```
The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; SUB3          METHOD 3, MODULE 2

;       NOTE THE DIFFERENCE BETWEEN THE MAIN-2.MAP AND THE MAIN-3.MAP

                PUBLIC SUBR3
SUB3_DATA       SEGMENT PUBLIC 'Z_DATA'
LOCAL_WORD      DW 0F372H
SUB3_DATA       ENDS

SUB_3_CODE      SEGMENT PUBLIC  'Z_CODE'
                ASSUME CS:SUB_3_CODE, DS:SUB3_DATA
SUBR3           PROC    FAR             ;FAR - CS REFERENCES THIS SEGMENT
                MOV     AX,LOCAL_WORD   ; AT ASSEMBLY TIME,
                JMP     NEXT_INSTR      ; SO A 2-WORD (SEG:OFFSET) CALL
NEXT_INSTR:     NOP                     ; MUST BE GENERATED ELSEWHERE.
                RET
SUBR3           ENDP
SUB_3_CODE      ENDS
                END
```

```
 Start  Stop   Length  Name                    Class
 00000H 00009H 000AH  MAIN_3_CODE              Z_CODE
 00010H 00017H 0008H  SUB_3_CODE               Z_CODE
 00020H 00023H 0004H  STACK
 00030H 00031H 0002H  SUB3_DATA                Z_DATA
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; MAIN-4          CONNECTING SEGMENTS AND MODULES

;   METHOD #4  -USING THE GROUP DIRECTIVE. LIKE METHOD #3, EXCEPT SEGMENTS IN
;     THE SAME GROUP GET CONCATENATED TO FORM ONE PHYSICAL SEGMENT AT LINK
;     TIME. ALLOWS THE SEGMENT REGISTERS TO REFERENCE A GROUP, WHICH IS 64K OR
;     SMALLER, MEANING 1-WORD CALLS AND OFFSETS ALL REFERENCE THE SAME PLACE.
  STACK           SEGMENT STACK   ;"STACK" AS SEGMENT COMBINE TYPE IS MAGIC.
                  DW 2 DUP(?)
STACK             ENDS

ALL_CODE          GROUP MAIN_4_CODE
MAIN_4_CODE       SEGMENT  PUBLIC 'Z_CODE'
                  ASSUME CS:ALL_CODE,SS:STACK
                  EXTRN SUBR4:NEAR          ;PUT THE EXTRN INSIDE THE SEGMENT
BEGIN_MAIN_4      PROC  FAR                 ; SINCE I KNOW IT IS IN ALL_CODE.
                  PUSH  DS          ;SET UP FOR RETURN TO DOS
                  XOR   AX,AX
                  PUSH  AX
                                    ;SUBR4 IS IN THIS PHYSICAL SEGMENT,IN ANOTHER
                  CALL  SUBR4       ; MODULE, ASSEMBLER USES OFFSET RELATIVE TO THE
                                    ; GROUP NAME, LINKER FIXES IT ALL UP.
                  RET               ;RETURN TO DOS
BEGIN_MAIN_4      ENDP
MAIN_4_CODE       ENDS
                  END BEGIN_MAIN_4


The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; SUB4            METHOD 4, MODULE 2

;       NOW WE CAN USE A NEAR PROCEDURE. THE DATA, OF COURSE, CAN BE IN A
;       GROUP TOO, BUT IN THIS EXAMPLE THERE IS ONLY ONE DATA SEGMENT.

                  PUBLIC SUBR4
SUB4_DATA         SEGMENT PUBLIC
LOCAL_WORD        DW (?)
SUB4_DATA         ENDS

ALL_CODE          GROUP SUB_4_CODE

SUB_4_CODE        SEGMENT PUBLIC  'Z_CODE'
                  ASSUME CS:ALL_CODE, DS:SUB4_DATA
SUBR4             PROC    NEAR
                  MOV     AX,LOCAL_WORD
                  JMP     NEXT_INSTR
NEXT_INSTR:       NOP
                  RET
SUBR4             ENDP
SUB_4_CODE        ENDS
                  END

 Start  Stop   Length  Name                    Class
 00000H 00007H 0008H  MAIN_4_CODE             Z_CODE
 00010H 00017H 0008H  SUB_4_CODE              Z_CODE
 00020H 00023H 0004H  STACK
 00030H 00031H 0002H  SUB4_DATA


The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; MAIN-5          CONNECTING SEGMENTS AND MODULES

; METHOD #5  -USING NEAR CALLS AND CLASSNAMES. LIKE METHOD #3, EXCEPT SEGMENTS
;       WITH THE SAME CLASSNAME GET CONCATENATED ON PARAGRAPH BOUNDARIES AT LINK
;       TIME. IF THE SEGMENT CONTAINING THE EXTERNAL REFERENCE IS KNOWN, PUT THE
;       EXTRN IN THE SEGMENT, OR USE A "DUMMY" SEGMENT IN THE CLASS OR GROUP.

STACK             SEGMENT STACK   ;"STACK" AS SEGMENT COMBINE TYPE IS MAGIC.
                  DW 2 DUP(?)
STACK             ENDS

MAIN_5_CODE       SEGMENT  PUBLIC 'Z_CODE'
                  EXTRN SUBR5:NEAR
                  ASSUME CS:MAIN_5_CODE,SS:STACK

BEGIN_MAIN_5      PROC  FAR
                  PUSH  DS          ;SET UP FOR RETURN TO DOS
                  XOR   AX,AX
                  PUSH  AX
                                    ;SUBR5 MAY BE IN THIS PHY. SEGMENT,IN ANOTHER
                  CALL  SUBR5       ; MODULE, ASSEMBLER TRUSTS WHERE THE EXTRN
                                    ; EXTRN STATEMENT IS.
                  RET               ;RETURN TO DOS
BEGIN_MAIN_5      ENDP
MAIN_5_CODE       ENDS
                  END BEGIN_MAIN_5


The IBM Personal Computer MACRO Assembler 02-11-83          PAGE    1-1

; SUB5            METHOD 5, MODULE 2

                  PUBLIC SUBR5
SUB5_DATA         SEGMENT PUBLIC 'Z_DATA'
LOCAL_WORD        DW 1234
SUB5_DATA         ENDS

SUB_5_CODE        SEGMENT PUBLIC  'Z_CODE'
                  ASSUME CS:SUB_5_CODE, DS:SUB5_DATA
SUBR5             PROC    NEAR
                  MOV     AX,LOCAL_WORD
                  JMP     NEXT_INSTR
NEXT_INSTR:       NOP
                  RET
SUBR5             ENDP
SUB_5_CODE        ENDS
                  END

 Start  Stop   Length  Name                    Class
 00000H 00007H 0008H  MAIN_5_CODE             Z_CODE
 00010H 00017H 0008H  SUB_5_CODE              Z_CODE
 00020H 00023H 0004H  STACK
 00030H 00031H 0002H  SUB5_DATA               Z_DATA

  Address         Publics by Value

 0001:0000       SUBR5

Program entry point at 0000:0000
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

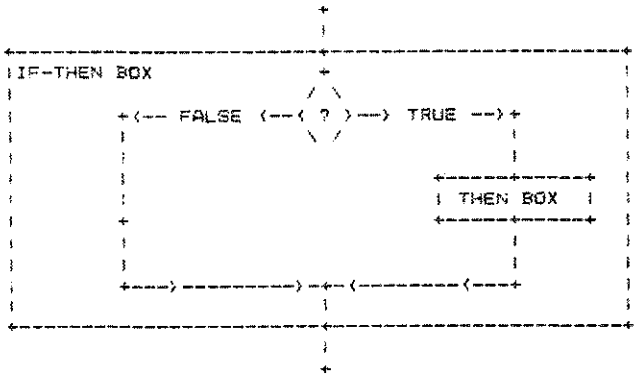
SOME SUGGESTIONS FOR CODING IN IBM PC/BASIC

PART 2

by Joe Rayhawk, Portland, OR

STRUCTURE: IF-THEN

This consists of the following: the entrance to the structure leads to code which evaluates some condition. If the condition is true, one BOX of code is entered. After execution of the internal BOX, control is passed to the bottom of the main BOX. If the condition is false, the control is passed to the bottom of the BOX.



Purpose or Use

Allows us to execute one set of code only when a condition is true. This is clearly just a degenerate form of the full IF-THEN-ELSE. It deserves separate mention because it is such a frequent pattern. If we did not allow it, we would need an explicit way to refer to a NULL BOX.

Example of BASIC Code - Many Minor Variations

```
110 IF condition THEN GOTO Statement-NBR
111 IF condition THEN Statement-NBR
112 IF condition GOTO Statement-NBR

125 IF condition THEN Statement
250 IF condition THEN
            Statement-A:Statement-B:Statement-C
```

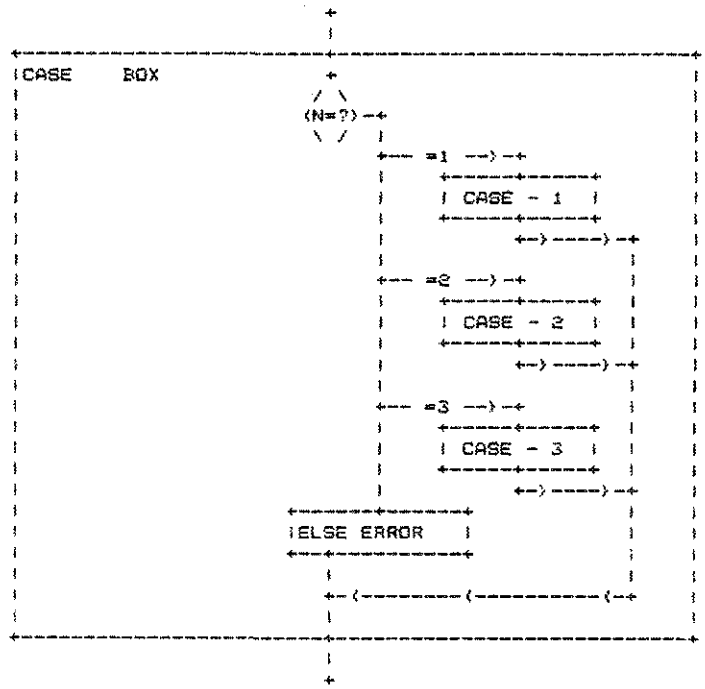
Several important points about the above examples.

First, Lines 110, 111 and 112 show several different syntaxes for accomplishing the same branch. These are historical remnants of various versions of BASIC. PC/DOS attempts to support all the old approaches here.

Second, Lines 125 and 250 illustrate actual use of regular statements as the THEN-BODY. Line 125 is fine as it stands. Line 250 may contain a surprise. Many programmers who read a line 250 for the first time expect that if the condition is false statement-A will be skipped and Statements-B and C will be executed. Such is not the case. The actual implementation on most micros is that condition being false causes control to be passed to the next line. In our terminology, Statements-A,B and C are part of an Internal THEN BOX. This implementation turns out to be a good decision for small programs and only a slightly bad decision for large programs.

STRUCTURE: CASE

This consists of the following: the entrance to the structure leads to code which evaluates an expression whose value is typically a small integer. This value is used as an index into a table of pointers to distinct blocks of code. One of these blocks is executed. After whichever block is executed, control is passed to the bottom of the structure.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Purpose or Use

Allows us to execute one of a set of code. One can look at a CASE structure as a series of cascaded IF-THEN-ELSE structures IF N = 1 THEN CASE-1 ELSE IF N = 2 THEN CASE-2 ELSE IF N = 3 THEN CASE-3 ELSE ERROR

In some systems, CASE structures are implemented internally as such cascaded IF-THEN-ELSEs. Most systems find ways which runs faster.

Example of BASIC Code - Two Important Variations

1023 ON number GOTO stmtnbr1,stmtnb2,...
2092 ON number GOSUB stmtnbr1,stmtnb2,...

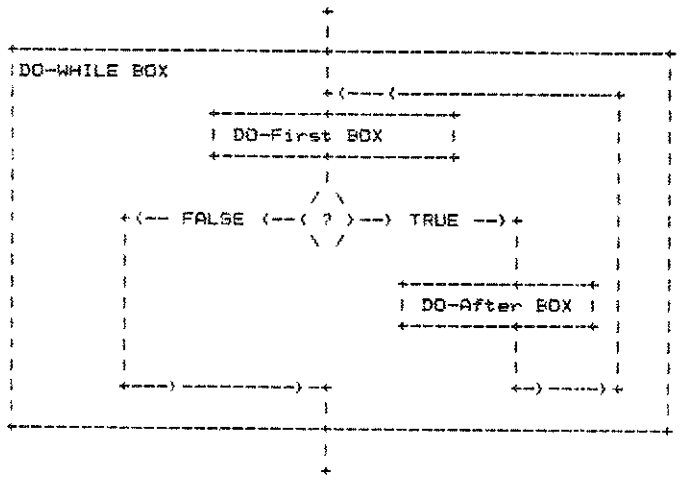
Notes:

If number is less than one or greater than the number of statement numbers, fall through. We might follow line 1023 with a line that processed the fall through as an error.

Line 2092 could not use that approach. In this case, we are calling for a GOSUB. The functioning of the GOSUB is to remember where we are, go off an begin executing at the required statement number. When the other code is finished, it returns control back to line 2092 via a RETURN command. Line 2092 will pass control on to the next line, say 2093. Hence, we cannot use the next line as an error catcher, when doing an ON ... GOSUB.

STRUCTURE: Generalized LOOP

This consists of the following: the entrance to the structure leads to an Internal BOX which does some process. This in turn leads to code which evaluates some condition. If the condition is true, an second internal BOX of code is entered. After this BOX is executed, control returns to the top of the first internal BOX which is re-executed. If the condition is false, control is passed out the bottom of the BOX.



Purpose or Use

This allows us to repeat a process that involves doing some preparatory process needed to be able to make a decision. If the decision is true, we have to do some further processing. After doing the further processing, we have repeat the preparatory process to discover if there is still further work to do.

This structure is frequently used when reading data from an external source and we are not finished until something external tells we are.

Example of BASIC Code:

```
1000 REM Example: READ and Sum AMOUNT Until NAME$ ) ID$
1010 REM Loop Point                    ' Loop Point.
1020   INPUT #1,NAME$,AMOUNT           ' Do-First
1030   IF(NAME$)ID$) GOTO 1500         ' Decision
1040   SUM = SUM + AMOUNT              ' Do-After
1050   GOTO 1010                       ' Loop to Top
1500 REM Continue                      ' Exit this Structure
```

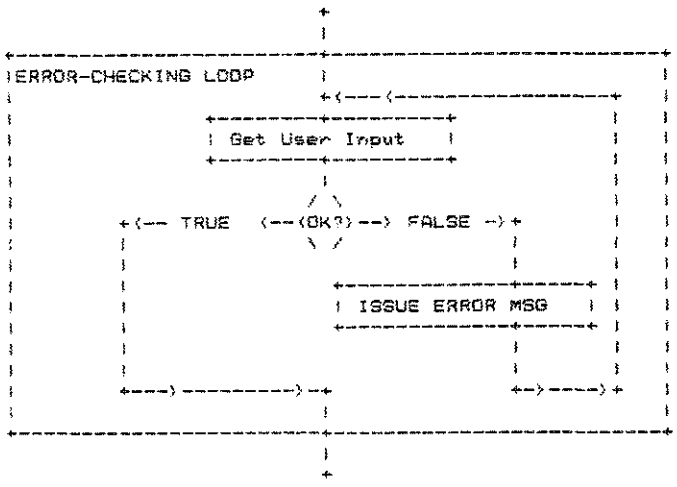
Notes:

BASIC does not have an explicit structure for supporting this. Usually, DO-FIRST code is some version of an INPUT Statement: READ, INPUT, LINE INPUT, INKEY$, GET.

STRUCTURE: Error-Checking Loop

This consists of the following: the entrance code consists of asking a user for some information. This information is then passed through one or more validity checks. If it is valid, then controls passes out the bottom of the structure. If it is invalid, an error message is issued and the user is re-prompted for the answer.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Purpose or Use

This allows us to formally define an approach to making sure that the user has given us valid information before we process the data and possibly hurt data in a file or worse. Hence, we can define the full interface to the user as a special form of the generalized loop in which we ask our question UNTIL we get a good answer. Otherwise, we issue an error message and loop back up.

An important aspect of this kind of loop which distinguishes it visually from a normal Generalized-DO is that we may have multiple validity checks. Each one may be able to issue an error message and branch up to the line which begin to prompt the user for the input. We could actually create artificial lines (perhaps with REMs) and stay within the strict limits of a series of generalized-DOs. However, the code is usually quite clear without going this far.

Example of BASIC Code:

```
1000 REM Get Student Name and Score
1010   PRINT:PRINT"Enter Student's Name and Score: ";
1020   INPUT  SNAM$,SCORE
1030   IF 20)LEN(SNAM$) GOTO 1040
1032     BEEP:PRINT
1034     PRINT" Sorry. I cannot handle more than 20 char"
1035     PRINT" Please re-enter with a shorter name."
1038     GOTO 1010
1040   IF 0(=SCORE AND SCORE(=100 GOTO 1050
1042     BEEP:PRINT
1044     PRINT" Sorry. SCOREs must be between 1 and 100."
1046     PRINT" Please re-enter with a VALID name."
1048     GOTO 1010
1050 REM OK, now Process Name and Score.
```

STRUCTURE Extension: SUBROUTINE

This consists of the following: A means exists that lets us switch control flow from its current point while remembering where we are. We then enter into a BOX called a subroutine BOX. The Subroutine BOX acts like one of our other BOXes, except: When it comes time for control to exit out the bottom of its BOX, it returns control to the remembered old execution point.

Effectively, the CALL SUBROUTINE and RETURN allows us to logically insert a BOX of code in line with our regular structures without having to actually build the code in that position. The advantages are basically those of saving space.

Purpose or Use

There are two important features of subroutine capabilities. First, it allows one copy of the code to be usable from multiple positions in a program. This saves space. Second, we can regard the subroutine as a more formal version of our standard BOX. If we have a well-defined and well-written subroutine we can view it as a black box that just does a service for us. While we are writing the code which calls the subroutine, we can concentrate on what it is trying to do and not on how the subroutine does its piece. Hence, the existence of the subroutine may allow us to concentrate on the main process' details better.

BASIC's Approach: GOSUB and RETURN Statements

Example: Note Line 1000... is the subroutine

```
100 MSG$="Y/N Do You Understand"
110 CHOICE$="Y,y,N,n"
120 GOSUB 1000
130 UND = ("Y"=KEYBD$) OR ("y"=KEYBD$)

200 MSG$="Select Function: 1=Add 2=Delete 3=Sort 4=Report"
210 CHOICE$="1,2,3,4"
220 GOSUB 1000
230 FUNC=VAL(KEYBD$)

1000 PRINT MSG$
1010 LINE INPUT KEYBD$
1020 IF (0=INSTR(KEYBD$,",")) GOTO 1040
1030   PRINT "ERROR. Your input must not include a comma"
1032   PRINT "Please Re-Enter."
1034   GOTO 1000
1040 IF(0()INSTR(CHOICE$+",",KEYBD$+",")) GOTO 1050
1042   PRINT "ERROR. Invalid choice. Choose one of:"
1044   PRINT CHOICE$
1046   GOTO 1000
1050 RETURN
```

Notes:

BASIC's GOSUB and RETURN implement the basic function needed for a subroutine, but just barely. In particular, we need to know the actual names of some of the variables used by the subroutine (MSG$, KEYBD$, CHOICE$). This is unfortunate. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BIOS EQUATES

ROM BIOS EQUATES

available from various BBS's

```
;---------------------------------
;   EQUATES
;---------------------------------
PORT_A          EQU     60H             ; 8255 PORT A ADDR
PORT_B          EQU     61H             ; 8255 PORT B ADDR
PORT_C          EQU     62H             ; 8255 PORT C ADDR
CMD_PORT        EQU     63H
INTA00          EQU     20H             ; 8259 PORT
INTA01          EQU     21H             ; 8259 PORT
EOI             EQU     20H
TIMER           EQU     40H
TIM_CTL         EQU     43H             ; 8253 TIMER CONTROL PORT ADDR
TIMER0          EQU     40H             ; 8253 TIMER/CNTER 0 PORT ADDR
TMINT           EQU     01              ; TIMER 0 INTR RECVD MASK
DMA08           EQU     08              ; DMA STATUS REG PORT ADDR
DMA             EQU     00              ; DMA CHANNEL 0 ADDRESS REG PORT ADDR
MAX_PERIOD      EQU     540H
MIN_PERIOD      EQU     410H
KBD_IN          EQU     60H             ; KEYBOARD DATA IN ADDR PORT
KBDINT          EQU     02              ; KEYBOARD INTR MASK
KB_DATA         EQU     60H             ; KEYBOARD SCAN CODE PORT
KB_CTL          EQU     61H             ; CONTROL BITS FOR KEYBOARD SENSE DATA
;---------------------------------
;  8088 INTERRUPT LOCATIONS
;---------------------------------
ABS0            SEGMENT AT 0
STG_LOC0        LABEL   BYTE
                ORG     2*4
NMI_PTR         LABEL   WORD
                ORG     5*4
INTS_PTR        LABEL   WORD
                ORG     8*4
INT_ADDR        LABEL   WORD
INT_PTR         LABEL   DWORD
                ORG     10H*4
VIDEO_INT       LABEL   WORD
                ORG     1DH*4
PARM_PTR        LABEL   DWORD           ; POINTER TO VIDEO PARMS
                ORG     01EH*4          ; INTERRUPT 1EH
DISK_PTR        LABEL   DWORD
                ORG     01FH*4          ; LOCATION OF POINTER
EXT_PTR         LABEL   DWORD           ; POINTER TO EXTENSION
                ORG     7C00H
BOOT_LOCN       LABEL   FAR
ABS0            ENDS
                PAGE
;---------------------------------
;   SYSTEM DATA AREA
;---------------------------------
BIOS_BREAK      DB      ?           ; BIT 7 = 1 IF BREAK KEY HAS BEEN DEPRESSED
RESET_FLAG      DW      ?           ; WORD = 1234H IF KEYBOARD RESET UNDERWAY
DATA            ENDS
;
;---------------------------------
;   EXTRA DATA AREA
;---------------------------------
XXDATA          SEGMENT AT 50H
STATUS_BYTE     DB      ?
XXDATA          ENDS
;
;---------------------------------
;   VIDEO DISPLAY BUFFER
;---------------------------------
VIDEO_RAM       SEGMENT AT 0B800H
REGEN           LABEL   BYTE
REGENW          LABEL   WORD
                DB      16384 DUP(?)
VIDEO_RAM       ENDS
;
;---------------------------------
;   ROM RESIDENT CODE
;---------------------------------
ROMCODE         SEGMENT AT 0F000H
                DB      57344 DUP(?)            ; FILL LOWEST 56K
                DB      '5700051 CORP. IBM 1981' ; COPYRIGHT NOTICE
ROMCODE         ENDS
;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;--------------------------------
;  STACK -- USED DURING INITIALIZATION ONLY
;--------------------------------
STACK           SEGMENT AT 30H
                DW      128 DUP(?)
TOS             LABEL   WORD
STACK           ENDS
;--------------------------------
;  ROM BIOS DATA AREAS
;--------------------------------
DATA            SEGMENT AT 40H
RS232_BASE      DW      4 DUP(?)        ; ADDRESSES OF RS232 ADAPTERS
PRINTER_BASE    DW      4 DUP(?)        ; ADDRESSES OF PRINTERS
EQUIP_FLAG      DW      ?               ; INSTALLED HARDWARE
MFG_TST         DB      ?               ; INITIALIZATION FLAG
MEMORY_SIZE     DW      ?               ; MEMORY SIZE IN K BYTES
IO_RAM_SIZE     DW      ?               ; MEMORY IN I/O CHANNEL
                PAGE
;--------------------------------
;  KEYBOARD DATA AREAS
;--------------------------------
KB_FLAG         DB      ?
;------ SHIFT FLAG EQUATES WITHIN KB_FLAG
INS_STATE       EQU     80H             ; INSERT STATE IS ACTIVE
CAPS_STATE      EQU     40H             ; CAPS LOCK STATE HAS BEEN TOGGLED
NUM_STATE       EQU     20H             ; NUM LOCK STATE HAS BEEN TOGGLED
SCROLL_STATE    EQU     10H             ; SCROLL LOCK STATE HAS BEEN TOGGLED
ALT_STATE       EQU     08H             ; ALTERNATE SHIFT KEY DEPRESSED
CTL_SHIFT       EQU     04H             ; CONTROL SHIFT KEY DEPRESSED
LEFT_SHIFT      EQU     02H             ; LEFT SHIFT KEY DEPRESSED
RIGHT_SHIFT     EQU     01H             ; RIGHT SHIFT KEY DEPRESSED
;--------------------------------
KB_FLAG_1       DB      ?               ; SECOND BYTE OF KEYBOARD STATUS
INS_SHIFT       EQU     80H             ; INSERT KEY IS DEPRESSED
CAPS_SHIFT      EQU     40H             ; CAPS LOCK KEY IS DEPRESSED
NUM_SHIFT       EQU     20H             ; NUM LOCK KEY IS DEPRESSED
SCROLL_SHIFT    EQU     10H             ; SCROLL LOCK KEY IS DEPRESSED
HOLD_STATE      EQU     08H             ; SUSPEND KEY HAS BEEN TOGGLED
;--------------------------------
ALT_INPUT       DB      ?               ; STORAGE FOR ALTERNATE KEYPAD ENTRY
BUFFER_HEAD     DW      ?               ; POINTER TO HEAD OF KEYBOARD BUFFER
BUFFER_TAIL     DW      ?               ; POINTER TO TAIL OF KEYBOARD BUFFER
KB_BUFFER       DW      16 DUP(?)       ; ROOM FOR 15 ENTRIES
KB_BUFFER_END   LABEL   WORD
;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
NUM_KEY         EQU     69              ; SCAN CODE FOR NUMBER LOCK
SCROLL_KEY      EQU     70              ; SCROLL LOCK KEY
ALT_KEY         EQU     56              ; ALTERNATE SHIFT KEY SCAN CODE
CTL_KEY         EQU     29              ; SCAN CODE FOR CONTROL KEY
CAPS_KEY        EQU     58              ; SCAN CODE FOR SHIFT LOCK
LEFT_KEY        EQU     42              ; SCAN CODE FOR LEFT SHIFT
RIGHT_KEY       EQU     54              ; SCAN CODE FOR RIGHT SHIFT
INS_KEY         EQU     82              ; SCAN CODE FOR INSERT KEY
DEL_KEY         EQU     83              ; SCAN CODE FOR DELETE KEY
                PAGE


;--------------------------------
;   DISKETTE DATA AREAS
;--------------------------------
SEEK_STATUS     DB      ?               ; DRIVE RECALIBRATION STATUS
;                                       BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE
;                                       NEXT SEEK IF BIT IS = 0
INT_FLAG        EQU     080H            ; INTERRUPT OCCURRENCE FLAG
MOTOR_STATUS    DB      ?               ; MOTOR STATUS
;                        BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING
;                        BIT 7   = CURRENT OPERATION IS A WRITE, REQUIRES DELAY
MOTOR_COUNT     DB      ?               ; TIME OUT COUNTER FOR DRIVE TURN OFF
MOTOR_WAIT      EQU     37          ; TWO SECONDS OF COUNTS FOR MOTOR TURN OFF
;
DISKETTE_STATUS DB      ?           ; SINGLE BYTE OF RETURN CODE INFO FOR STATUS
TIME_OUT        EQU     80H         ; ATTACHMENT FAILED TO RESPOND
BAD_SEEK        EQU     40H         ; SEEK OPERATION FAILED
BAD_NEC         EQU     20H         ; NEC CONTROLLER HAS FAILED
BAD_CRC         EQU     10H         ; BAD CRC ON DISKETTE READ
DMA_BOUNDARY    EQU     09H         ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
BAD_DMA         EQU     08H         ; DMA OVERRUN ON OPERATION
RECORD_NOT_FND  EQU     04H         ; REQUESTED SECTOR NOT FOUND
WRITE_PROTECT   EQU     03H         ; WRITE ATTEMPTED ON WRITE PROT DISK
BAD_ADDR_MARK   EQU     02H         ; ADDRESS MARK NOT FOUND
BAD_CMD         EQU     01H         ; BAD COMMAND PASSED TO DISKETTE I/O
;
NEC_STATUS      DB      7 DUP(?)    ; STATUS BYTES FROM NEC
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

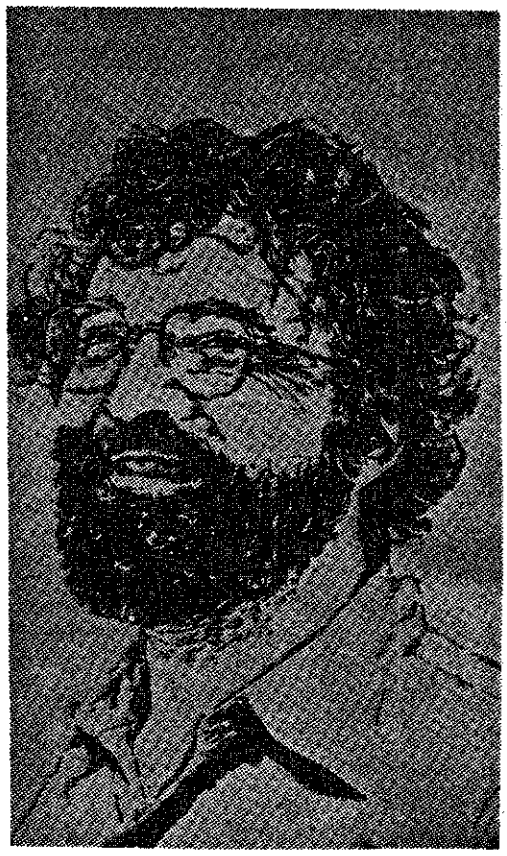
```
;-----------------------------------
;   PROGRAM SEGMENT PREFIX
;-----------------------------------
;
PROGRAM         SEGMENT
INT20           DW                      ;INT 20 for DOS return
MEMORYSIZE      DW                      ;TOTAL MEMORY SIZE
CALLDOS_FAR     DB      5 DUP(?)        ; LONG CALL DOS DISPATCHER
TERMADDR        DB      4 DUP(?)        ; TERMINATE ADDRESS (ip,cs)
CTRLBREAK       DB      4 DUP(?)        ; ctrl_break EXIT ADDRESS
RESERVED        DB      4ah DUP(?)      ; RESERVED AREA
FCB_ONE         DB      10H DUP(?)      ; unopened fcb #1
FCB_TWO         DB      10H DUP(?)      ; unopened fcb #2
PARM_LENGTH     DB                      ; LENGTH OF PARAMETERS
PARM_FIELD      DB      19 DUP(?)       ; PARAMETER STRING DATA
;
                SUBTTL
                PAGE
                ORG     100H            ; DEFAULT ENTRY ADDRESS
PROGRAM_ENTRY   EQU

;-----------------------------------
;   VIDEO DISPLAY DATA AREA
;-----------------------------------
CRT_MODE        DB      ?           ; CURRENT CRT MODE
CRT_COLS        DW      ?           ; NUMBER OF COLUMNS ON SCREEN
CRT_LEN         DW      ?           ; LENGTH OF REGEN IN BYTES
CRT_START       DW      ?           ; STARTING ADDRESS IN REGEN BUFFER
CURSOR_POSN     DW      8 DUP(?)    ; CURSOR FOR EACH OF UP TO 8 PAGES
CURSOR_MODE     DW      ?           ; CURRENT CURSOR MODE SETTING
ACTIVE_PAGE     DB      ?           ; CURRENT PAGE BEING DISPLAYED
ADDR_6845       DW      ?           ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
CRT_MODE_SET    DB      ?           ; CURRENT SETTING OF THE 3X8 REGISTER
CRT_PALLETTE    DB      ?           ; CURRENT PALLETTE SETTING COLOR CARD
                PAGE
;-----------------------------------
;   TIMER DATA AREA
;-----------------------------------
TIMER_LOW       DW      ?           ; LOW WORD OF TIMER COUNT
TIMER_HIGH      DW      ?           ; HIGH WORD OF TIMER COUNT
TIMER_OFL       DB      ?           ; TIMER HAS ROLLED OVER SINCE LAST READ
;COUNTS_SEC     EQU     18
;COUNTS_MIN     EQU     1092
;COUNTS_HOUR    EQU     65543
;COUNTS_DAY     EQU     1573040 = 1800B0H
```


Disk Explorer
for your IBM/PC

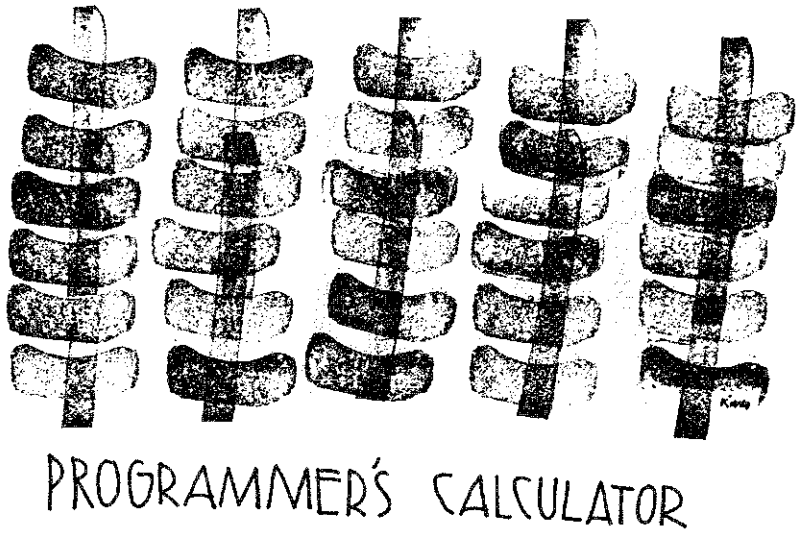
- goes well below D.O.S.
- can read/write any sector
- is not stopped by copy-protect schemes
- recognizes D.O.S. Formats (1.0, 1.1, 2.0)
- recognizes 10-SECTOR JFORMAT

package includes 18-page manual



Available from computer stores or for $100 from:
Richard Larratt and Associates Ltd.
RR #1 Demorestville
Ontario, CANADA K0K1W0
Phone 613-476-5309

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PROGRAMMER'S CALCULATOR

by Ronald Moak

One of the hassles of programming in assembly language is having to deal with the conversion of numbers between hexadecimal, decimal and binary. After fighting it for a month with pencil and paper, I decided that I needed something faster and more accurate. There are a couple of good hand held calculators on the market, but they cost money and I already have a computer. I decided instead to write a program to do the conversions on the computer.

The conversion program allows you to input either a hexadecimal, decimal, binary, or octal number, and convert it to the other numbering systems. The program also allows you to calculate the two's complement of any number between -32,768 and 32,767. The conversion will work on any 20 bit number to allow for the full range of the PC's memory addressing.

After completing the conversion program, I needed a calculator program. The result was a calculator that emulates the Hewlett Packard brand of calculators. The calculator uses Reverse Polish Notation logic and displays four stack registers and a display register on the screen. This allows you to view the contents of the registers without having to write them down. In addition to the working registers, six memory registers are displayed on the screen. All calculations are performed and displayed in double precision for greatest accuracy.

The calculator contains a number of standard functions such as add, subtract, multiply, divide, square root, square, Y to the x, Y to the 1/X, inverse and absolute value. The four main functions (+,-,*,/) may be performed on any number within memory without disturbing any number in the stack. A number of other functions were included that deal primarily with programming. The calculator will allow you to perform AND, OR, XOR, EQU, IMP, and NOT logic operations on or between numbers.

The two programs were combined into one program and may be toggled between with a function key. When you switch between programs, the one you leave is frozen where you left. This allows you to leave the calculator to perform a conversion and return without disturbing the calculator.

Programmer's Calculator is being offered under the Freeware philosophy of distribution. A copy of the program and documentation may be obtained by sending a formatted disk and self-addressed disk mailer with postage to:

Ronald Moak
Forest Computer Consultants
P.O. Box 1463
Springfield, OR 97477
ph. 503-746-7470

If you do not have an extra disk, then send $5.00 to cover the disk and handling. The recommended contribution if you like the program is $15.00. I hope you will find this program as useful to you has is it has been to me. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BARGRAPH PROGRAMS FOR dBASE II

by Daniel Ridder

Financial planners and home accountants have turned to graphs and charts to provide added meaning to numerical data. Although dBASE II does not directly support graphics, it can easily manipulate IBM's extended character set to produce useful and interesting bargraphs which can be displayed on a graphics or an IBM monochrome monitor.

A bargraph is produced by stacking the ASCII block characters - CHR(219), CHR(176), CHR(177), or CHR(178) - in a given column. The height of this stack is controlled by an accumulator loop which repeats a block character a certain number of times according to a value derived from three factors: values extracted from the datafile records; values chosen for the y-axis range; and, the number of rows used by the graph.

For example, to graph a $135 phone bill using a 20-row graph with a y-axis range of $100 to $200, each row represents a $5 increment: $200-100 divided by 20. Subtracting the lower limit ($100) from the $135 bill leaves $35, and 35 divided by $5 is the loop value. Here, the block character repeats for seven rows.

Following this article are two program listings illustrating this technique. These programs use a checkbook or accounts payable data file. The first program BARGRAPH.PRG computes the monthly payments for two selected years of a selected payee and displays the results in a 20-step bargraph.

The datafile's record structure should resemble this one:

```
FIELD NAME,TYPE,WIDTH,DECIMAL PLACES

001 PAYEE,C,30
002 DATE,C,8
003 AMT,N,8,2
004 CODE,C,4
```

Other fields may be added. BARGRAPH.PRG only assumes that payment amounts are stored on a field named AMT and the date is stored in conventional (mm/dd/yy) form on a field named DATE. The PAYEE field contains Payee's name and the CODE field contains Payee categories.

BARGRAPH.PRG uses the FIND command because it gives faster results than the SUM command. Thus, the field containing the selected Payee's identity must be INDEXED. For economy, consider indexing only part of the PAYEE field, e.g.,:

```
.INDEX ON $(PAYEE,1,12) TO XPAYEE
```

Sometimes it is preferable to display each record of a selected Payee rather than totalling by month (e.g., the May and June phone bills both paid in June). The second program GRAPHSEQ.PRG displays the data from 24 chronological records of a selected Payee beginning at a selected date.

GRAPHSEQ.PRG uses the same datafile structure and an indexed payee identity field. In addition, to ensure that consecutive records are chronological, the DATE field must be INDEXED and used as a secondary index, as follows:

```
.INDEX ON $(DATE,7,2)+$(DATE 1,5) TO XDATE

.USE (filespec) INDEX XPAYEE,XDATE
```

Parts of the following two programs are similar but each one uses date and amount values differently. GRAPHSEQ.PRG gives faster results, but for real speed consider putting data and index files on a ram disk. □

INTERNATIONAL PC OWNERS

IPCO

"The international user group for the PC"

- Software library of 175+ programs @ $3 each
- Datacount® General Ledger/Manual $35
- Volkswriter® $125
- Bimonthly newsletter 40+ pages
- Worldwide information exchange
- "Freeware"

FOR FREE CATALOG WRITE:

INTERNATIONAL PC OWNERS, INC.
ATTN: FREE CATALOG
P.O. BOX 10426
PITTSBURGH, PA 15234

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
* - BARGRAPH.PRG
* - Monthly totals of selected ITEM graphed
* - ITEM field must be INDEXED
* - Database must use fieldnames AMT & DATE
* with date in conventional (mm/dd/yy) form

* A) Define environment & variables
SET TALK OFF
SET INTENSITY OFF
SET COLON OFF
STORE 0 TO A,B,C,X,Y,ROW,XAMT
STORE 'AAJanFebMarAprMayJunJulAugSepOctNovDec' TO MONTH
STORE CHR(219)+CHR(219) TO BAR
STORE CHR(177)+CHR(177) TO BAR2
STORE 25 TO GRAFCOL
STORE 1 TO Z,STOP

* B) Keyboard entries
ACCEPT "Select ITEM (eg. Joe's Bar)" TO ITEM
ACCEPT 'Enter FIELD name where ITEM located (eg. PAYEE)' TO XFIELD
? ' THIS FIELD MUST BE INDEXED!'
? '      Enter years to graph - Two digits only:'
ACCEPT 'Enter Current year (eg. 83)' TO YR
ACCEPT 'Enter prior year (eg. 82)' TO Y2
? '      Enter numerical spread for vertical axis:'
INPUT 'Enter upper limit(eg. 100)' TO A
INPUT 'Enter lower limit (eg. 0)' TO B
ACCEPT 'Enter y-axis label: PCT. or DOL.' TO LABEL
ACCEPT 'Enter top-line Title (45-char. max.)' TO TITLE

* C) Clear screen, add labels
ERASE
@ 24,35 SAY 'Monthly Totals of '+ITEM
@ 23,45 SAY BAR+'19'+YR
@ 23,27 SAY BAR2+'19'+Y2
@ 23,21 SAY LABEL
@ 0,30 SAY TITLE

* D) Generate 20-step y-axis labels
STORE (A-B)/20 TO C
DO WHILE ROW <21
 STORE 1+ROW TO ROW
 @ ROW,15 SAY A
 STORE A-C TO A
ENDDO
```



```
* E) Draw edge along y-axis labels
* - Create LINE variable here too
STORE CHR(95)+CHR(95) TO LINE
STORE 0 TO ROW
DO WHILE ROW < 24
 STORE 1+ROW TO ROW
 STORE LINE+CHR(95)+CHR(95) TO LINE
 @ ROW,25 SAY CHR(186)
 @ ROW,19 SAY CHR(186)
ENDDO

* F) Display screen lines
STORE 1 TO ROW
DO WHILE ROW <22
 @ ROW,26 SAY LINE
 STORE ROW+4 TO ROW
ENDDO
STORE 1 TO ROW

* G) Main Loop - Heart of program
STORE !(ITEM) TO ITEM
FIND &ITEM
STORE T TO MORE
DO WHILE MORE

* H) Find ITEM, one month at a time
  IF &XFIELD='&ITEM'
* -- Go to next record if date condition
* not met - Eg. when Z=1, Date=January
    IF .NOT.(VAL($(DATE,1,2))=Z.AND.$(DATE,7,2)=YR)
      SKIP
      LOOP
    ENDIF
* -- Add AMT if date condition met
    STORE AMT+XAMT TO XAMT
    SKIP
    LOOP
  ENDIF
* -- Display month (from MONTH string) &
* -- AMT totals on left side of screen
  STORE $(MONTH,(Z*3),3) TO XMONTH
  @ ROW,9 SAY XAMT
  @ ROW,3 SAY XMONTH+YR
  STORE ROW+1 TO ROW

* I) Adjust amount so each bar increment
* equals 1/20th of y-axis range variation
  STORE (XAMT-B)/C TO X
  IF X>20
   STORE 20 TO X
  ENDIF
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
* J) Put bar at row 21, use variable to
* shift bargraph 4 columns per month
  STORE 21-X TO Y
  STORE Y TO GROW
  STORE GRAFCOL+4 TO GRAFCOL

* K) Graph using amount as loop variable
  DO WHILE GROW < X+Y
    STORE 1+GROW TO GROW
    @ GROW,GRAFCOL SAY BAR
  ENDDO

* L) Display x-axis labels once
  IF .NOT.STOP >12
    @ 22,(GRAFCOL-1) SAY XMONTH
  ENDIF

* M) Add 1 to Z & find following month
  FIND &ITEM
  STORE 0 TO XAMT
  STORE Z+1 TO Z

* N) After 12 months change variables
* - year, bar-type, reset column, etc.
  IF Z>12
    STORE Y2 TO YR
    STORE BAR2 TO BAR
    STORE 23 TO GRAFCOL
    STORE 1 TO Z
    STORE STOP + 12 TO STOP
  ENDIF

* O) End - enter <return>
  IF STOP >24
    SET CONSOLE OFF
    WAIT
    SET CONSOLE ON
    STORE F TO MORE
  ENDIF

* - After each month's pass, loop back
ENDDO
```

PAGE 3 dBASE II BARGRAPH.PRG (Graph monthly totals)

```
* - GRAPHSEQ.PRG
* - Graphs Consecutive Records of ITEM
* - ITEM & DATE FIELDS MUST BE INDEXED
* INDEX DATE as $(DATE,7,2)+$(DATE,1,5)
* - Database must use Fieldnames AMT and
* DATE (in conventional (mm/dd/yy) form)

* A) Define Environment & variables
SET TALK OFF
SET INTENSITY OFF
SET COLON OFF
STORE 0 TO A,B,C,X,Y,ROW,XAMT
STORE 'AAJanFebMarAprMayJunJulAugSepOctNovDec' TO MONTH
STORE CHR(219)+CHR(219) TO BAR
STORE CHR(177)+CHR(177) TO BAR2
STORE 23 TO GRAFCOL
STORE 22 TO LABELROW
STORE 1 TO Z,STOP

* B) Keyboard entries
ACCEPT 'Select ITEM to find (eg. Ma Bell)' TO ITEM
ACCEPT 'Enter name of FIELD where ITEM located (eg. PAYEE)' TO XFIELD
? '  DATE FIELD & THIS FIELD MUST BE INDEXED!'
ACCEPT 'Graph records from date (mm/dd/yy):' TO START
STORE $(START,7,2)+$(START,1,5) TO START
? '     Enter numerical spread for vertical axis:'
INPUT 'Enter upper limit(eg. 100)' TO A
INPUT 'Enter lower limit (eg. 0)' TO B
ACCEPT 'Enter y-axis label: DOL. or PCT.' TO LABEL
ACCEPT 'Enter top-line title (45-char. max)' TO TITLE

* C) Clear screen, add labels
ERASE
@ 24,40 SAY 'Consecutive Records of '+ITEM
@ 23,21 SAY LABEL
@ 0,30  SAY TITLE

* D) Generate 20-step y-axis labels
STORE (A-B)/20 TO C
DO WHILE ROW <21
 STORE 1+ROW TO ROW
 @ ROW,15 SAY A
 STORE A-C TO A
ENDDO
```

PAGE 1 / dBASE II GRAPHSEQ.PRG (Graph consecutive records)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
* E) Draw edge along y-axis labels
*   Create LINE variable here too
STORE CHR(95)+CHR(95) TO LINE
STORE 0 TO ROW
DO WHILE ROW < 24
 STORE 1+ROW TO ROW
 STORE LINE+CHR(95)+CHR(95) TO LINE
 @ ROW,25 SAY CHR(186)
 @ ROW,19 SAY CHR(186)
ENDDO

* F) Display screen lines
STORE 1 TO ROW
DO WHILE ROW <22
 @ ROW,26 SAY LINE
 STORE ROW+4 TO ROW
ENDDO
STORE 1 TO ROW

* G) Main Loop - Heart of program
STORE 1(ITEM) TO ITEM
FIND &ITEM
STORE T TO MORE
DO WHILE MORE

* H) Find Item, Stop after 24 records
  IF &XFIELD='&ITEM'.AND..NOT.STOP >24
* - - Ignore if before start date
    IF $(DATE,7,2)+$(DATE,1,5) < '&START'
      SKIP
      LOOP
    ENDIF
* - - If after start date, store data
    STORE DATE TO XDATE
    STORE AMT TO XAMT

* I) Use month digit to get MONTH string
    STORE VAL($(DATE,1,2)) TO XMO
    STORE $(MONTH,(XMO*3),3) TO XMONTH
    @ ROW,9 GET XAMT
    @ ROW,0 GET XDATE
    STORE ROW+1 TO ROW

* J) Adjust amount so each bar increment
*  equals 1/20 of y-axis range variation
    STORE (XAMT-B)/C TO X
    IF X>20
      STORE 20 TO X
    ENDIF
```



```
* K) Put bar on row 21, use variables to
* shift bargraph 4 col after each record
    STORE 21-X TO Y
    STORE Y TO GROW
    STORE GRAFCOL+4 TO GRAFCOL

* L) Display x-axis labels on two rows
    @ LABELROW,GRAFCOL SAY XMONTH

* M) Graph using amount as loop variable
    DO WHILE GROW < X+Y
      STORE 1+GROW TO GROW
      @ GROW,GRAFCOL SAY BAR
    ENDDO


* N) After 12 months, change variables
* bar-type, reset month, column, etc
    STORE Z+1 TO Z
    IF Z>12
      STORE BAR2 TO BAR
      STORE 25 TO GRAFCOL
      STORE 1 TO Z
      STORE STOP + 12 TO STOP
      STORE 23 TO LABELROW
    ENDIF

* O) End - enter <return>
  ELSE
    SET CONSOLE OFF
    WAIT
    SET CONSOLE ON
    STORE F TO MORE

* P) After each pass, loop back
  ENDIF
  SKIP
ENDDO
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PROKEY

by Bill Hailey

I have been using PROKEY for a couple of weeks now and I like it very much. It gets rave reviews almost everywhere and it should. It is a very powerful package. Many products earn this distinction, but PROKEY seems to work with everything. For example, you are in the middle of COMPUSERVE and have to enter

RF 14xxx

(After G PCS-131 and QSN for quick scan new). Everything you want to see in the 14000's. So you want to Read Forward. But why type it when you have PROKEY??? Alternate equals activates PROKEY, type Function key 10 (for example) to assign the string to F10. Then RF 14 Alternate dash (next to equals). The cursor will change to show that you're in prokey while defining. Because you didn't press enter the string is on the screen. Enter the last 3 numbers and press return. From now on you just press F10. Of course, you could put this in a batch file to be executed with, say, PC-TALK or whatever you use.

When I boot my computer I want to use one of several topics. It might be assembly language or "C" or communications. I use PROKEY to give me the choice of copying the files I want to Electronic Disk. Prokey allows several lines with several enter statements so you can enter copy several times. Alt-D can be directory, Alt-T can be type, etc.

CHOICE.PRO

```
(alt=)(altc)copy a:clib.cl c:(Enter)
copy a:cc.bat c:(enter)
copy a:cl.exe c:(enter)
copy a:stdio.h c:(enter)
(alt-)
*
(alt=)(F8)copy a:go.bat c:(Enter)(alt-)
*
(alt=)(F9)prokey dos.pro/r(Enter)(alt-)
*
(alt=)(F10)date 8-(ctrl])  (ctrl])-83(Enter)(alt-)
*
(alt=)(altb)del *.bak(Enter)
(alt-)
*
(alt=)(altd)dir (alt-)
*
(alt=)(altt)type (alt-)
*
(alt=)(altp)a:prokey a:(alt-)
*
(alt=)(altm)b:(enter)comm(enter)(alt-)
*
(alt=)(alta)c:(Enter)w a:autoexec.bat(enter)(alt-)
*
(alt=)(F7)c:(Enter)w a:choice.pro(Enter)(alt-)
```

You get different cursors depending on what you're doing. Prokey allows both variable and fixed fields. This is very important. Say you don't have a real time clock. Prokey allows ASCII files.

```
<alt=><F10>date  8-<ctrl]> <ctrl]>-83
                                <Enter><alt->
```

The <> shows a special key being pressed. This allows two digits to be included, then prokey will fill in the rest. A variable field could be used but enter would have to be appended to the field.

ERROR MESSAGES

Prokey gives decent error messages but they could be better. I'd personally like the line number of the error reported if possible. Prokey gives the line in question, an error message, ("You're making an endless loop") and an asterisk under the questionable command.

The other problem is that Prokey is slow. You type a command for Prokey to filter, like Alt-D or Function 10, and wonder if you've got the right file. It will take a second for Prokey to respond. Which can be a long time when typing. I have version 2.12.

I like prokey a lot. I do wish it was faster. Perhaps in later versions. □

BAD.PRO

```
(alt=)(f1)(ctrlk)s(ctrlq)p(ctrlk)c(f1)(enter)b:(f1)(enter)(alt-)
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SOFTWARE REVIEWS

EDIX/WORDIX

by Morton F. Kaplon

In one of your recent issues you reviewed a variety of Text Editors. One that you omitted and which I recently obtained a copy of is EDIX (and its accompanying Text Formatter WORDIX) by Emerging Technology. EDIX sells for $195 and it is really something else. Let me say a few words about it. First of all, for context, I have owned and used VEDIT for quite some time and have found it a first class Text Editor. Its very compact code (about 15K) and flexibility are tremendous assets consequently I do not use EDLIN at all. EDIX, though being fatter codewise (about 65K) does just about everything that VEDIT does and some extremely useful things that VEDIT does not.

EDIX is command driven with the exception of a few well placed prompts in response to commands. The commands, in the format you get it, are all input via the Alt-Letter combination, with the exception of two that use Ctrl-d and Ctrl-f, four which use Alt-number and one which uses Alt-F10. The cursor is of course driven from the number pad as well as the Ctrl-cursor combinations.Some of the commands using Alt-letter and Alt-number are duplicated on the function key pad. The commands are extremely well chosen mnemonically and one learns them very quickly as you do the entire system. There is practically nothing one cannot accomplish. The system includes a very powerful set of macro commands that allow complete redefinition of the keyboard as well as the ability to extend the utilities included. For instance, I have completely redefined the command input so that all of the commands are contained on the function pad leaving the Alt-letter combination free for ProKey. Also added were some editing commands (e.g. delete to end of line) as well as moving some commands to the cursor pad that were not there for some inexplicable reason (cursor to end and beginning of line).

The really unique and powerful aspect of EDIX is the existence of 12 text buffers, each of which may contain a text file and the ability to display any combination of 1, 2, 3, or 4 of them simultaneously on the screen; to edit them in a group and to move and copy text between any of the four. Since EDIX allows you to access the disk directory, to append files from the disk onto any file in any buffer, and to write any buffer back to disk, with a file name of your choice, you can see that you have a really powerful utility. One moves easily from the edit screen to EDIX's command line where you can, as noted above, access the disk directory and also change file names and erase files, all using DOS commands. You can write a given file to disk and remain in Edit. If you attempt to leave EDIX with a buffer that has not been written to disk, you are notified that you have an unwritten buffer and prompted for a Yes or No to exit.

Copying and moving files is easily done. Text to be copied or moved is defined by two marks (the second one on prompt) and then copied or moved by another command. There is a very powerful (and fast) search and search and replace utility (here you get prompts for the search and replace strings). The latter has an extensive set of wild card options. There is a complete HELP menu callable from EDIT, though after using EDIX a few times you hardly ever need it. A very useful STATUS screen can be called from EDIT that shows the status of each of the 12 text buffers, as well as of the memory available for text. EDIX utilizes up to 250K of memory for Editing, shared between the 12 text buffers and it is very fast with one exception. That exception is reformatting which is relatively slow compared to performance on other commands. The screen scrolls to 256 characters and word wrap can be set up to that limit. Carriage return splits the line at the point of the cursor; I find that a very useful way of doing things. Normal edit mode is INSERT but OVERTYPE is available by using the Ins key and is flagged on the status line as well as the cursor changing from an underscore to half full. Tabs can be set from the Command line as well as included in the configuration file. Adjustment for color or black or white is also included in the configuration file with special flexible formats for your use.

The status line contains the complete filespec of the buffer being edited, the buffer ID, the column number of the cursor, the line number of the cursor and the total number of text lines in the buffer. In addition the various prompts and error messages are displayed there. The macro facility contained is very powerful. You can test your macros from the EDIT screen and include them,if desired, in a configuration file. When EDIX is called from disk, it loads at the same time a file called CONFIG.EDX which contains any particular macros or setup commands you desire. Thus, if you wanted different keyboard layouts for different usages, you would just construct different config.xxx files, renaming the one you want to use at a particular instant as config.edx

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

before calling EDIX.These files are very easy to construct and in this sense, EDIX is easier to configure than VEDIT is and not subject to VEDIT's inability to tolerate errors in configuration.

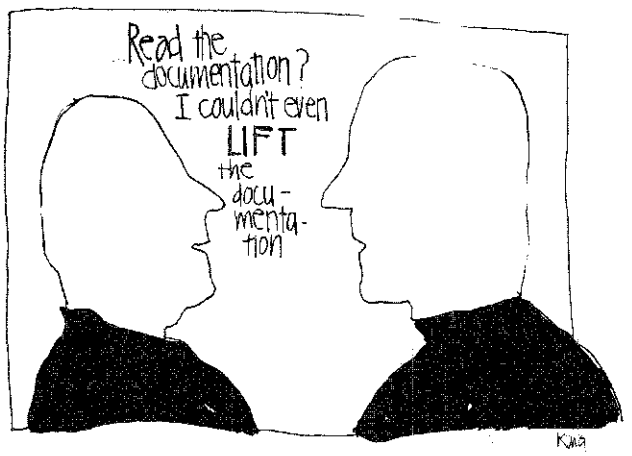
One can also include printer initialization codes in the text files of EDIX and it is very easy to construct macros that allow you to print an entire text file, or any marked portion of it just by using a single key. That is a reflection of the macro power of EDIX in which you can assign macros to any of the function keys, most of the Ctrl-key combinations,and to available Alt-key combinations (the reason why I redefined the KeyBoard). The macro commands include any keystroke command defined in EDIX as well as printer initialization codes and may be nested ten deep. The number available is limited by memory that is shared with the text buffers.

This Text Editor is a distinct pleasure to use and I cannot emphasize how useful the split screen facility is. I have found that in editing text it is very useful to make a copy of the text being edited to another screen so that direct comparisons between the original and the edited text are immediately available. These facilities make the copy and merge facilities of Word Processors seems like childs play.

As I noted, there is also available, as a separate package, a TEXT FORMATTER called WORDIX, also selling for $195.00. This is an extremely powerful utility, comparable in performance to text formatters available on UNIX facilities. It is not the sort of facility that one needs for ordinary correspondence, though it can of course do that very well, but using it for that is like using Fortran to add a column of numbers. Incidentally, since EDIX also embeds control characters (ASC 1-31) in text, you can construct, using the MACRO facilities, quite a bit of built in text formatting using the SEARCH commands to take you to specified locations and other macros to things there, such as headers or footers, insert white space, etc.

I mentioned earlier ProKey. This is perhaps one of the most useful utilities I have ever acquired and I include it on just about every disk I use. What it does, and does extremely well, is to allow immediate macro definition utilizing practically any function key or Alt-key keystroke from the keyboard (or you can also write them). Since the program allows nesting of macros as well as space for KB input, it gives you an enormous amount of flexibility. For instance, a trivial but useful application is the following. I have PROKEY on every disk that I boot, so set up that I am prompted to enter Ctrl-F1 to enter the date. This is so set up with ProKey that the date entered is then available from Ctrl-F2. I then have macros set up with ProKey to automatically enter the date on a variety of other utilities that I use. The inclusion of the date on everything I write happens to be one of my hangups. I have set up a Disk Directory system which uses the SORT utility of DOS 2.0 and macro commands called from ProKey. It is much faster and less grinding than using .bat files. If you haven't tried it yet, do, you will like it.

Finally a few words on some really trivial things that some may find convenient. I happen to like to use the function keys for all special keystrokes. If I can't manage to define them using the facilities available with a given program, then I can usually manage to define them using ProKey. This then gives rise to the question of how to make the definitions readily available. I have found the following most helpful. I buy those spiral card inserts for wallets. They contain five plastic inserts, 2.5'' by 3.75'' on a spiral edge. I then cut them in half so that each unit is 1.25'' by 3.75'' and make a holder to contain the function key pad definitions for each of the various programs.I enclose a sample of the holder and of the format(for EDIX as an example) that I use. I attach it by double sticky tape adjacent to the function pad and flip back and forth as different programs are in use. It is much faster and editable than those overlays, which really for the most part interfere with the key action. (The text is printed in Compressed, double strike to make it small enough). □



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BASIC DEVELOPMENT SYSTEM

Reviewed by Ron Moak

BASIC Development System
Softool Systems
8972 E. Hampton Ave. Suite 179
Denver, CO 80231
ph. 303-793-0145

The BASIC Development System (BDS) by Softool Systems contains seven utilities that may be accessed at BASIC's command level. These utilities expand the number of commands available to you during program development by offering functions that will find key words and strings in a program, provide a cross reference listing of the program, greatly expand BASIC's renumbering command, provide a single step trace for debugging programs, dump all program variables and their contents, and compress and uncompress program files.

During installation, BDS program modules are linked to BASIC.COM files. When BASIC is entered, the BDS is automatically loaded, allowing instant access to BDS's utilities. BDS provides two modes of operation depending upon how much memory your system has. Under disk resident mode, the program modules are kept on the disk and accessed when needed. This requires that a disk with BDS program modules remain in the drive at all times. Under memory resident mode, the program modules remain in the drive at all times. Memory resident requires approximately 11K of overhead for the modules.

BDS provides a single letter shorthand for entering basic commands. For example, clear screen may be invoked by typing C from the command line instead of CLS. A program listing may be made by entering L instead of typing LIST. Other commands have been added to ease listing programs. PgUp will list the previous program line and PgDn lists the next program line.

Single step trace is one of the most useful features provided. A program may be executed a line at a time with the current line displayed in the corner of the screen. The trace may be run on the whole program, or on selected portions that contain those unexplained logic errors. A trace may be halted at any point using Ctrl-Break, and a variable dump invoked to list all the variables and their current values. Variable dumps may include all of the variables or selected variables. Dumps are made to either the screen or the printer.

An expanded cross-reference utility creates a listing of all the variable names, line numbers and constants to screen or printer. By stipulating parameters in the command, a modified list may be created. For example, a program may be searched, and all of the references to a particular variable may be listed, or a listing of all variables starting with a particular variable. FIND command may be used to search for all remarks, key words, or strings. FIND allows you to replace one key word with another. Similar to the cross-reference, FIND will list all line numbers that reference a string or key word.

BDS's renumbering utility expands BASIC's RENUM function. BDS's RENUM allows you to set upper and lower limits on the blocks of lines to be renumbered. Unlike BASICS's RENUM function, BSD's RENUM function allows you to renumber blocks of code to fit within other blocks of code, provided there is sufficient gap in the line numbers for the new lines to fit. Blocks of code may be duplicated with only the line numbers changed. At the end of an input session, one renum command will scan the program to look for lines that were for some reason undefined, without changing the text. To prevent you from getting into serious trouble, error messages are displayed for invalid operations.

The compression utility will remove remarks, unnecessary blanks, extraneous colons, LET key words, redundant GOTO when combined with THEN GOTO or ELSE GOTO, non-executable lines, variable names from NEXT statements, and quotation marks at the end of a line. Most of the actions are controlled by parameters passed when the command is invoked. If desired, you can compress the program and leave the remarks intact. BDS claims compressed programs will occupy 30 to 40 percent less memory space, and will run 7 to 10 percent faster than uncompressed programs.

Uncompress reverses the process of the compress utility; however, it cannot restore comments or anything deleted from the original file. Under the compress utility FOR/NEXT and WHILE/WEND loops are indented in the new file for easier reading, multiple statements are split into separate lines and spaces are inserted into program code.

BDS's 40 pages of documentation is clear and concise, and should be no problem for even the novice user. My only complaint (a minor one), is that a memory map is needed in the documentation to illustrate where BDS's code resides in memory. A couple of programs I wrote which contained calls to assembly language subroutines located outside of BASIC, locked up the machine, because they overlayed BDS's program modules.

Overall, BASIC Development System is an excellent set of utilities that will at least make BASIC program development easier, if not easy. The package is well worth the investment if you are developing programs in BASIC. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ LIBRARY

PJ LIBRARY

UNIX REVIEW

A bimonthly magazine for UNIX has been introduced from Review Publications. Anything that applies to UNIX is included; users notes, hardware, software, reviews, groups etc. $18 per year

UNIX REVIEW
Review Publications
2711 76th Ave. SE
Mercer Island, WA 98040
ph. 206-232-6719

SOFTWARE NEWSLETTER

The Micro Software Letter discusses software news and trends in the market place. Business applications are covered, reviews of current products, as well as books. Write for a sample issue. ($124.49 per year)

Micro Software Letter
25 NOrth Broadway
Tarrytown, NY 10591
ph. 914-332-4545

TELEPHONES/COMPUTERS

If telecommunications via your Hayes is of primary interest then look into MODEM NOTES. A single source foe information about modems, online data bases, and the world accessed by telephone.

MODEM NOTES
PO Box 408472
Chicago, IL 60640
ph. 312-764-7407

DATA BASED ADVISOR

The new magazine exclusively for dBASE II users is finally out. dBASE source code programs are shown as well as explanations describing the constructs. Product reviews etc.

Data Based Solutions
1975 5th Ave. Suite 105
San Diego, CA 92101
ph. 619-236-1182

WHY YOU NEED COPY II PC:

1. COPY II PC allows you to backup your protected software. And unlike other backup programs, COPY II PC makes exact copies without modifications to the software on the duplicate disk. This assures maximum reliability and compatibility with the widest range of protection schemes. COPY II PC handles sector timing, multiple sector sizes, and bad sector ID's as used for disk protection verification.

2. COPY II PC is a complete replacement for "DISKCOPY" on your PC-DOS diskette. It formats, copies, then verifies the duplicate disk (DISKCOPY does not verify). COPY II PC automatically uses all the memory in your PC, and adjusts for single/dual sided diskettes. A drive speed utility helps keep your drives in top condition. For your convenience, COPY II PC is not copy-protected.

Copy II PC works with your IBM PC and most compatibles.

CENTRAL POINT Software, Inc.
P.O. Box 19730-203
Portland, OR 97219
(503) 244-5782

$39.95
(Plus $2 s/h)

Attention Apple owners: Call us about backing up your protected software with Copy II Plus!

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LOCAL AREA NETWORK

Architecture Technology Corp has put together a "Designers Handbook" for local area networks. The first part of the book covers LAN design issues and the last ¾ of the book lists all of the known LAN hardware-software suppliers and the features contained in each (Baseband, Broadband, and Voice/Data PBX). 325 pages in length.

Architecture Technology Corp.
PO Box 24344
Minneapolis, MN 55424
ph. 612-935-2035

PC WEEK

Ziff-Davis, besides venturing into the DISK-Magazine business, will be publishing a PC Week newspaper style periodical. It looks a little like Computer News or Computer World, a weekly with a business bent into the DP/MIS management level of information. A subscription is free if you wish to add it to your list. You must include your company name, your title, and answer 10 questions about your use of computers and software to fill up someones database somewhere. (It would be nice if every company in the US listed their DP machinery once and it was put in an appropriate national data base so we all wouldn't have to answer the questions over and over again.) More mainframe and "paperless" articles. (Free.)

PC Week
PC Subscriptions
PO Box 445.
Boulder, CO 80321

EUROPEAN MAGAZINE

Europe has it's own IBM PC publication called the PC USER. It is published from London. The issue we saw was well put together and had several interviews touching sensitive topics, like what DID happen when IBM asked Gary Kildail about using CP/M for the PC. Gary doesn't usually mention the word Microsoft, he always refers to them as the 'competition" , but Bill Gates is quite at home talking freely about anyone and never seems to balk about calling DRI by name.

PC USER
57a Hatton Garden
London, UK
EC1N 8JD
ph. 01-837-3699

ROBOTICS

The Robotics Society of America has their own journal. They publish it bimonthly and it is sent to all of its members. Student $15, Personal $25, Corporate $100.

Robotics Society of Am.
200 California Ave. Suite 215
Palo Alto, CA 94306

SOFTWARE MARKETING HANDBOOK

There is a new book out from Software Research that provides a list of markets for software publishers. It is called MICRO SOFTWARE DISTRIBUTORS and is available from:

Software Research Co.
Drawer A
PO Box 9524
Washington, DC 20016
ph. 202-364-8700

The price is $95 dollars plus $4 dollars for shipping.

Approximately 170 companies are profiled who distribute software. Each distributor is featured on one page and the description includes the kinds of software it seeks, what it pays in royalties, the marketing rights it prefers, who it sells to, the support it provides, how to submit a proposol, etc. The companies included are hardware firms, software houses, book publishers, hardware and software distributors, mail-order houses, retailers, and other organizations actively seeking programs for resale.

MICROSOFT ROYALTY POLICIES

Microsoft Corp has joined several other companies in dropping its run time royalty payments. Previously you were required to make payment for programs that were compiled using (R) BASIC, Business BASIC, and COBOL. These fees have been dropped. Their other languages, C, FORTRAN, and Pascal did not have such requirements. Do note that this is for the Microsoft versions of the compilers, and not the IBM marketed software.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MISC

SCREEN SAVER

If you are concerned about leaving your system on when you go for some coffee and don't make it back in time to turn the intensity down (think about leaving the Personal Editor up for long periods of time with the bottom line in reverse video,) then you might be interested in Logical Systems SCREEN SAVER. It is a stay resident routine that blanks your screen after a user defined period of time. You return to the keyboard, strike any key and the image returns. Also on the diskette are other preventative maintenance programs. One tests memory, another tests the disk, and a third keeps track of the number of hours a diskette has been used.

SCREEN SAVER ($19.95)
Logical Systems Corp.
Route 1 Box 253
St. Michael, MN 55376
ph. 612-497-3861

BLUE BOOK SOFTWARE MARKETING

There is a new approach to marketing your software taken by BABY BLUE BOOK Inc. You will receive up to 50% royalties for letting them sell your programs. They only need a non-exclusive marketing agreement. They publish a large book of IBM PC software offerings. The whole book is made up of software they are promoting. When a request to purchase a particular product is given to them, they send out the diskette (and documentation on diskette) that you have supplied to them. They are essentially an advertising distributor for you. For more information contact:

The BABY BLUE BOOK, Inc
Submission Dept.
P.O. Box 70188
Marietta, GA 30007

SAL/86

Protools provides a Structured Assembly Language compiler for programmers doing a lot of complex ASM maintenance. Contact:

PROTOOLS
24225 Summerhill Ave.
Los Altos, CA 94022

PROGRAM GENERATOR

Key Solutions has announced a program generator and full screen processor tool for the PC called "DATABURST" . Besides the usual creation of a BASIC program, it also provides an ASM subroutine for keyboard input. Screens are easily changed and you can be editing up to 16 screens at a time. The product is written in assembly for speed. ($225)

Key Solutions, Inc.
PO Box 2297
Santa Clara, CA 95055
ph. 408-554-6125

BASIC PRODUCTIVITY TOOLS

KEYTOOLS includes a number of productivity tools for programmers. "PICASSO" is a screenmaker, "EMERSON" is a formatted data entry subroutine, "SOCRATES" builds online help facilities, and "YOUNGMAN" is a set of one-liners that you can use in your own code. ($40)

KEY-1 Computer Systems
178 Spring St.
Newport, RI 02840
ph. 401-849-4562

SELL YOUR SOFTWARE

ATC has put together a report on "HOW TO SELL YOUR SOFTWARE" It covers gaining access to directory listings, being included in data bases, receiving media coverage, writing a users manual, and running a successful mail order program. For a copy ($20) write:

ATC Software
RT 2 Box 448
Estill Springs, TN 37330

KERMIT

This is a communications program that runs on various micros and mainframes. It is available for more systems then is listed in the flyer reprinted here, and there is a media charge for supplying the product since the demand has been so high.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

11/82

LCG
Third Party Software

Kermit

KERMIT is a protocol for transferring files between computers over telecommunication lines. It does packetizing, checksumming, and retransmission to ensure data integrity. Microcomputer implementations of KERMIT also provide terminal connection at any speed allowed by both the micro and the host. KERMIT is like many other such protocols, but with several differences:

- It's free.
- It's documented.
- It's well tested and in wide use.
- Implementations exist for many different systems, including DEC and IBM mainframes and many microcomputers. Each implementation is written in the language best suited for a particular machine or operating system.

KERMIT was developed at the Columbia University Center for Computing Activities primarily to facilitate offline storage of DECSYSTEM-20 files on microcomputer floppy disks. The first implementations were for the DECSYSTEM-20, IBM VM/CMS systems, and 8080/Z80-based microcomputers running CP/M. The KERMIT specification has since served as the basis for new KERMITs. Any KERMIT can communicate with any other KERMIT.

Here's a list of known KERMITs as of November, 1982:

KERMIT-20	TOPS-20, native mode. Written in MACRO-20.
KERMIT-10	TOPS-10, adapted from KERMIT-20 at Stevens Institute of Technology.
KERMIT-C	For any system that has a C compiler. Versions have been successfully built for PDP-11 and VAX UNIX systems.
KERMIT-CMS	IBM VM/CMS systems on machines with the 370 instruction set. Written in assembly language. Does ASCII/EBCDIC conversion.
KERMIT-80	For 8080 and Z80 based micros running CP/M. Written in 8080 assembler. VT52 emulation is provided unless otherwise indicated. Minor customizations (mainly the specification of serial port addresses, and details of screen control) are required for specific manufacturers. The following are presently available:

- DEC PC100 "Rainbow" operates as VT100.
- DEC VT-180 "Robin", operates as VT100.
- Intertec SuperBrain.
- Apple II with Z80 Soft Card and D.C. Hayes Micromodem II.
- Heath/Zenith-89. Operates as H19 terminal.
- Vector Graphics.

KERMIT-86	For the IBM Personal Computer running PC DOS. Written in 8086 assembler. VT52 emulation.

All implementations listed above, with the exception of KERMIT-C, provide a TOPS-20-style command parser, including keyword recognition and built-in help. KERMIT-20 uses the COMND JSYS, KERMIT-10 uses GLXMAC/ORNMAC, KERMIT-80 and KERMIT-86 simulate the COMND JSYS, KERMIT-CMS simulates COMND as much as is possible on a half-duplex system. Work is in progress on other implementations, and on improvements in current ones.

KERMIT does not rely on any host system software or servers. All actions are initiated explicitly by the user. Typically, the user starts KERMIT on a microcomputer, "connects" through KERMIT to the remote host, logs in, starts KERMIT on the remote host, "escapes" back to the micro, then sits back and watches the packets fly. KERMIT ensures that packets will be sent no faster than the host can receive them by waiting for an acknowledgement of receipt before sending the next packet. KERMIT can transfer files singly or in wildcard groups. KERMIT can be used between mainframes when appropriate connections exist.

HOST REQUIREMENTS:

KERMIT assumes the following:

1. All hosts can communicate in ASCII. KERMITs running on non-ASCII hosts are responsible for character set conversion.
2. All printable ASCII characters (in the range 40-176 octal) are acceptable as input to the host and will not be transformed in any way.
3. A single nonprintable ASCII character can be used for packet synchronization. The character is normally Control-A (SOH, ASCII 1).
4. If a host requires a line terminator for terminal input, that terminator must be a single ASCII character, such as CR or LF.
5. The host's terminal input buffer is at least long enough to receive the longest ACK packet (about 15 characters).
6. If a host requires padding, the padding character is in the range ASCII 0-37 or ASCII 177 (octal).
7. If KERMIT is to transfer binary files, both communicating hosts must be capable of 8-bit terminal communication, i.e. they must be able to ignore parity.

KERMIT does not assume:

1. Anything about baud rate.
2. That the host can do XON/XOFF or any other kind of flow control. This kind of flow control can be initiated behind KERMIT'S back by commands to the host computers. If the hosts support any kind of flow control, then it can be used to cut down on retransmission due to buffering problems.
3. That the host is capable of full duplex operation. Any mixture of half and full duplex hosts is supported.

In addition, KERMIT does not assume that the host has the ability to time out; if neither host participating in a KERMIT connection can do so, then KERMIT allows for manual intervention to wake up protocol deadlocks.

ORDERING INFORMATION:

The complete KERMIT distribution tape may be obtained free of charge from Columbia University by sending a magnetic tape and a cover letter telling what machines and operating systems KERMIT will be used on. A return mailer would be appreciated. 9-track tapes can be produced at 800, 1600, or 6250 bpi in TOPS-20 DUMPER format, TOPS-10 BACKUP format, or in IBM EBCDIC OS standard label or CMS format. Please indicate the desired tape format. The tape includes source and binary (or hex) for all implementations of KERMIT, plus the KERMIT manual and specification document. A printed copy of the document is also included. Send requests to:

KERMIT Distribution
Columbia University Center for Computing Activities
7th Floor, Watson Laboratory
612 West 115th Street
New York, N.Y. 10025

If you make any changes to KERMIT, or produce a new version, you are encouraged to send your work (including documentation) to the above address, so that other KERMIT users might benefit from it. Full credit will be given to contributors.

LCG Third Party Software Bulletins are published by Applications Marketing, Large Computer Group, Marlboro, MA.

The information provided herein is for the independent evaluation of users of the DECsystem-10 and DECSYSTEM-20 computers. Digital has not undertaken an investigation as to the operability or functionality of the products herein and does not endorse any such products. Digital disclaims all obligations with respect to the accuracy of the statements and the performance or the maintenance of the products contained herein. Digital is not responsible for any errors which may appear in this bulletin.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	*EduSystem*	*RSTS*
DECsystem-10	*IAS*	*RSX*
DECSYSTEM-20	*KA10*	*TOPS-10*
DECUS	*KL10*	*TOPS-20*
DECnet	*MASSBUS*	*UNIBUS*
DIBOL	*OMNIBUS*	*VMS*
DIGITAL LOGO	*PDP*	*VAX*

CP/M is a registered trademark of Digital Research, Inc.
UNIX is a registered trademark of Bell Laboratories
VM/CMS is a registered trademark of International Business Machines Corporation
EBCDIC is a registered trademark of International Business Machines Corporation

digital

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SPF EDITORS

There are at least three companies that sell IBM mainframe SPF style editors. We will be reviewing them next issue. The ones we know of are:

Rogue River Software
2822 Tahitian Ave.
Medford, OR 97501
ph. 503-779-3002

PHASER Systems
24 California St.
San Francisco, CA 94111
ph. 415-434-3990

Digital Research Inc.
160 Central Ave.
Pacific Grove, CA 93950

UNETIX

Lantech Systems has a multi = tasking operating system that is compatible with UNIX software and is designed for networking of micros. It contains an MS-DOS emulator to run both MS-DOS and UNIX applications. A single user can have up to 10 windows and transfer data from one to the other. The operating systems runs on 8086/8088 based micros and is being ported to the Motorola 68000 and National 16032 microprocessors. For additional information:

Lantech Systems
9625 Wendell Road
Dallas, TX 75243
ph. 214-340-4432

HONOR TEXT EDITOR

Basic Business Software is offering a full screen text editor (SCREEN) under the HONORWARE system (very much like Freeware). You send in $5 and receive the text editor and manual (on diskette). If you like the product you can then send in $35 and receive the manual in a 3 ring binder and updates. The editor works with Microsoft's Mouse, thus allowing an easy and inexpensive way to try out a mouse driven text editor.

SCREEN-Text Editor
Basic Business Software
PO Box 26311
Las Vegas, NV 89126

BASIC LISTING SCROLL PRODUCT

If you are tired of typing "LIST xxx-xxx" when you are in BASIC then IRC Assoc has the answer. Their product takes care of all the scrolling deficiencies in IBM BASIC. You can LIST and stop the scroll with one key, or relist with one key. The RENUM function is enhanced also. ($20)

ONE-KEY STEP-SCROLL
IRC Associates Dept K
125 Sherrfield Dr, X9
Saginaw, MI 48603
ph. 517-792-2156

THE ULTIMATE PC FAIRE

Finally the ultimate PC Faire is happening in Houston, Texas during February 2-5, 1984. It is billed as the IBM World Exposition for buyers and sellers of IBM PC Products. Very soon we will all have national IBM PC Faires in our local neighborhoods.

Mitch Hall Assoc.
PO Box 860
Westwood, MA 02090
ph. 617-329-7466

HP TERMINAL EMULATION

If you have an HP at the office and want to plug in your PC then Inner Loop Software can provide the emulators. The emulator is compatible with software for HP2624, HP2648, and VT52 type terminals. ($200)

Inner Loop Software
PO Box 45857-C
Los Angeles, CA 90045
ph. 213-645-5162

6-PAK TO GO

AST has perhaps the ideal board for PC's. It is their 'SixPakPlus" board. Six rows of memory chips are provided to bring a new PC up to the max of 640K. ALso the normal clock, one parallel port, one serial port, and optional game adapter are provided. What more can you need? Certainly an excellent product by a leader in the field. Available from your local dealer.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

WE KNOW YOU'RE SERIOUS ABOUT THE IBM PC.

SO ARE WE!

WE KNOW WHAT YOU WANT.

- *Detailed how-to articles*
- *Programming tips*
- *In-Depth product reviews*
- *Problem reporting and solutions*
- *Code listings*

WE'VE ALWAYS HAD IT ...

Some features in previous issues were a technical look at the system unit, number crunching, utility programs, asynchronous communications, monitor and printer character sets, personalizing DOS, BASIC vs. Pascal, menu generation programs, unleashing your printer, evaluation of hard disk drives, and plotting data with PC graphics.

WE ALWAYS WILL.

Features for 1983 will include extensive information about networks, database managers, spreadsheets, operating systems, the 8087 chip, professional program development, and much more.

WE KNOW WHICH MAGAZINE YOU NEED ...

PERSONAL COMPUTER AGE

The Definitive Journal for the IBM Personal Computer User

CALL TOLL FREE: **800-824-7888 operator 77**
California only: 800-852-7777 op. 77

I NEED WHAT YOU'VE GOT! **ENTER MY SUBSCRIPTION TODAY!**

☐ I WANT INFORMATION ABOUT BACK ISSUES, TOO!

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY:

☐ VISA CARD NO. _____

☐ MASTERCARD EXPIRATION DATE _____

AUTHORIZED SIGNATURE _____

☐ 1 yr. (12 issues) $24.00
☐ 2 yrs. (24 issues) $45.00
☐ 3 yrs. (36 issues) $64.00
☐ NEW ☐ RENEWAL

FOREIGN SUBSCRIPTIONS
1 YR. (12 ISSUES)
CANADA/MEXICO $29.00

OTHER COUNTRIES,
WRITE FOR RATES
PAYMENTS MUST BE
IN U.S. FUNDS DRAWN
ON A U.S. BANK

NAME _____

ADDRESS _____

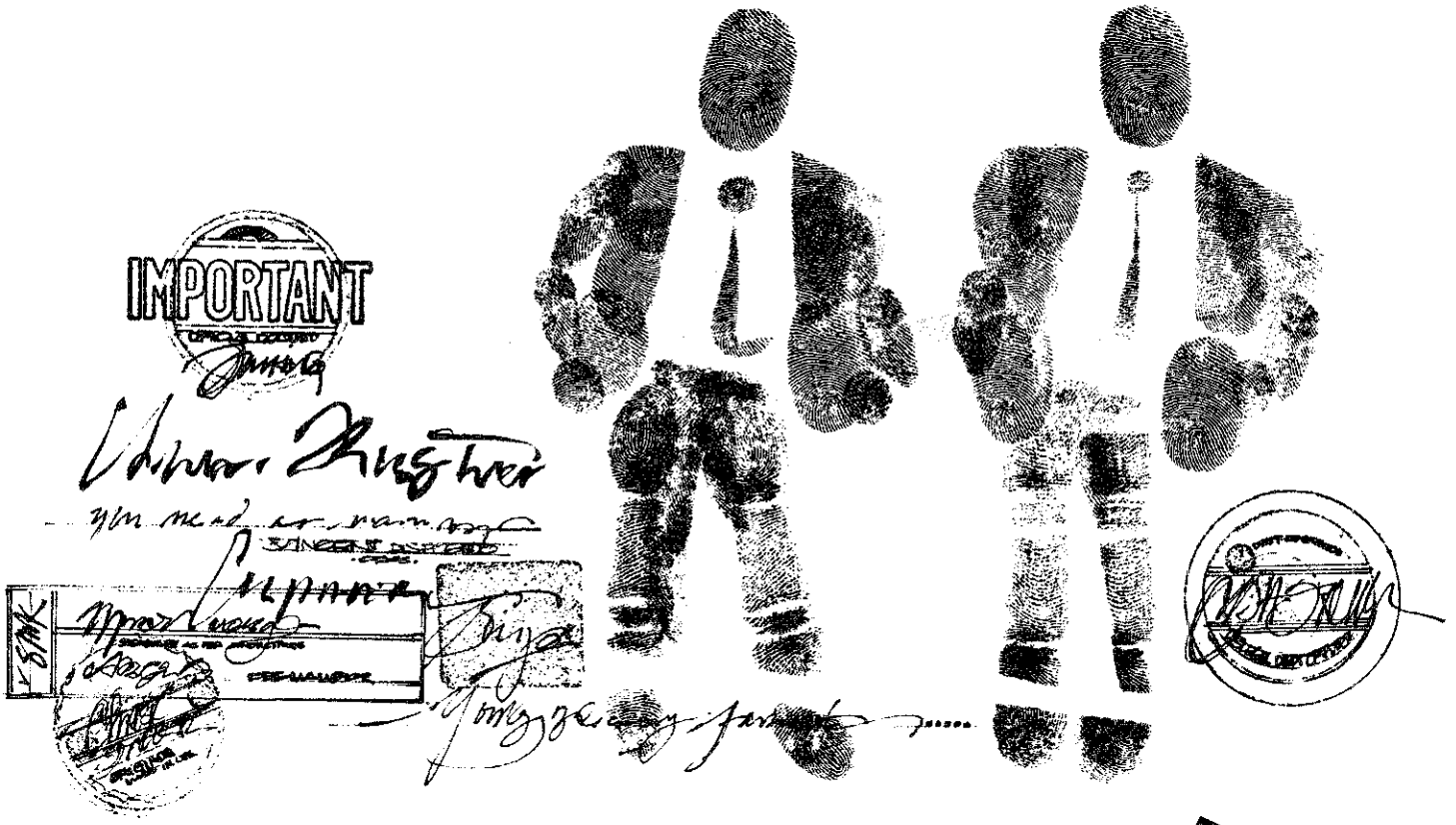
CITY _____

STATE _____ ZIP _____

Personal Computer Age

Department PJ5
P.O. Box 70725
Pasadena,
California 91107

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



You can reinvent the wheel or you can subscribe to the Programmer's Journal.

Vol. 1 #1: Norton Pascal I/O, Program style manual, COBOL, System Analysis, BASIC modules, Rounding error.
Vol. 1 #2: Defining character sets, C86 BIOS calls, Copy protection, Pascal code, Input routines, Writing an Assembler.
Vol. 1 #3: ASM dumb terminal, C source code, ASM graphics line drawing, Text editors reviewed, utilities.
Vol. 1 #4: Diskette directory via Pascal, MASM WHILE-END, C/ASM interface, ASM stay resident routine, structured programming.

SUBSCRIPTION

Include your:

Name _____

Company _____

Address _____

City _____ State _____

Zip _____

Languages you use: _____

Languages interested in: _____

1 year bi-monthly subscription rate:
$24 U.S. ($30 Canada, $40 other)
☐ Check enclosed (all back issues sent)
☐ Bill me

Begin my subscription with:
☐ First issue
☐ Current issue

Send to:
PROGRAMMER'S JOURNAL
2765 Potter St.
Eugene, OR 97405
Ph. (503) 687-0398

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED